ISO TC184/SC4/*_WG3_ N_277_ (P_1__)
*Complete with EC (for Editing), PMAG, or WG

Date: 29 April 1994 **Supercedes SC4/ WG3 N 277p (P 1)**

PRODUCT DATA REPRESENTATION AND EXCHANGE

Part: 204 **Title:** Application protocol: Mechanical design using boundary representation

Purpose of this document as it relates to the target document is:

____ Primary Content
____ Issue Discussion
____ Alternate Proposal
____ Partial Content

Current Status: Draft CD

ABSTRACT:
This Part of ISO 10303 specifies an application protocol for mechanical design using boundary representation (B-rep) shape models. It combines 3 types of B-rep model, one for polyhedron models, one for elementary models and one for more general B-rep models into a single AIM. AICs are used extensively in documenting the AIM.

KEYWORDS:
Application Protocol, B-rep, Mechanical Design, AIC

Document Status/Dates (dd/mm/yy)

Part Documents		Other SC4 Documents	
18 May 94	Released		Released
	Project		Working
29 Apr 94	Working		Editorial OK
8 May 94	Technically Complete		Technically Complete
17 May 94	Editorially Complete		Approved
	ISO Committee Draft		

Owner/Editor: Ray Goult
Address: 33 Filgrave,
Newport Pagnell,
Bucks.
MK16 9ET.
England
Telephone/FAX: +44 234 711653 (+Fax)
E-mail:

Alternate:
Address:
Telephone/FAX:
E-mail:

Comments to Reader
This is an edition of Part 204 CD prepared in response to the integration workshop in Long Beach December 1993 and qualification workshop at NIST March 94. It is based entirely upon the published DIS versions of resource Parts and will be updated after the completion of the CD ballot to conform to the final IS versions of Parts 41-46.

Contents

Annexes

Figures

Tables

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liason with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

International Standard ISO 10303–204 was prepared by Technical Committee ISO 184, *Industrial automation systems and integration,* Subcommittee 4, *Industrial data and global manufacturing programming languages.*

The preparation of this document has benefitted from the technical contributions of many projects and their sponsoring organizations. The contributions of the following are acknowledged:

- Esprit project 2195 CADEX
- Esprit project 6040 Prodex

This is the first edition of this part of ISO 10303.

Annexes A, B, C and D are an integral part of this part of ISO 10303. Annexes E, F, G, H, J and K are for information only.

This part is one in a series of parts which together form the International Standard ISO 10303, Industrial automation systems and integration – Product data representation and exchange. At the time of publication of this part of ISO 10303, the following parts had been registered for international ballot:

- Part 1 Overview and fundamental principles;
- Part 11 Description methods: The *EXPRESS* language reference manual;
- Part 21 Implementation methods: Clear text encoding of the exchange structure;
- Part 31 Conformance testing methodology and framework: General concepts;
- Part 32 Conformance testing methodology and framework: Requirements on testing laboratories and clients;
- Part 41 Integrated generic resources: Fundamentals of product description and support;
- Part 42 Integrated generic resources: Geometric and topological representation;

- Part 43 Integrated generic resources: Representation structures;
- Part 44 Integrated generic resources: Product structure configuration;
- Part 45 Integrated generic resources: Materials;
- Part 46 Integrated generic resources: Visual presentation;
- Part 47 Integrated generic resources: Shape variation tolerances;
- Part 49 Integrated generic resources: Process structure and properties;
- Part 101 Integrated application resources: Draughting;
- Part 104 Integrated application resources: Finite element analysis;
- Part 201 Application protocol: Explicit draughting;
- Part 202 Application protocol: Associative draughting;
- Part 203 Application protocol: Configuration controlled 3D design of mechanical parts and assemblies;
- Part 207 Application protocol: Sheet metal die planning and design.

The reader may obtain information on the other Parts of ISO 10303 from the ISO Central Secretariat.

Introduction

ISO 10303 is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a neutral mechanism capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases, and archiving.

ISO 10303 is organized as a series of parts, each published separately. The parts of this International Standard fall into one of the following series: description methods, integrated resources, application protocols, abstract test suites, implementation forms, and conformance testing. The series are described in ISO 10303–1. This part of ISO 10303 is a member of the Application protocol series.

An application protocol provides a scope and a context for the general-purpose constructs defined in the integrated resources class of parts. Since adequate conformance testing can only be based on a valid set of requirements, implementations can only be tested in terms of an application protocol.

This Part of ISO 10303 specifies an application protocol(AP) for mechanical design using boundary representation solid models.

This application protocol defines the context, scope, and information requirements for mechanical design using boundary representation models and specifies the integrated resources necessary to satisfy these requirements.

Application protocols provide the basis for developing implementations of ISO 10303. Application protocols provide the basis for developing abstract test suites for the conformance testing of AP implementations.

Clause 1 defines the scope of the application protocol and summarizes the functionality and data covered by the AP. An application activity model that is the basis for the definition of the scope is provided in annex E. The information requirements of the application are specified in clause 4 using terminology appropriate to the application. A graphical representation of the information requirements, referred to as the application reference model, is given in annex F.

This Part of ISO 10303 contains the definition of conforming boundary representation solid models and the mechanisms to transfer them via an exchange structure as defined in Part ISO 10303-21. In this Part B-reps are characterised by the fact that they can represent models with only planar surfaces (facetted B-rep), models with only analytical surfaces (elementary B-rep) and Models with sculptured surfaces and curves (advanced B-rep). The application reference environment in which these B-rep models are used is the generation and exchange of volume based data in the Computer-aided Mechanical design process. This application places fundamental requirements on the model exchange and the neutral representation of models. The transfer and archiving of B-rep models at different stages of the design and engineering process requires the following to be maintained:

– the completeness of the models when mapped between application systems;

– the correctness of semantics of the representation;

– the accuracy of the relationships between B-rep model entities and instances.

Three different classes of implementation are specified in clause 6.

This application protocol was developed as one component of a series of Mechanical Design application protocols and is complemented by ISO 10303-205 Mechanical Design Using Surface Models. These Parts share a common application environment and have a similar scope for the representation of mechanical parts. The significant differences among these Parts of ISO 10303 is in the manner in which the shape of a mechanical part is represented. In this Part the representation is as a manifold solid boundary representation model. In ISO 10303-205 the shape of the part is represented by a surface model in which all surfaces and bounding curves are fully represented. Figure 1 gives a pictorial representation of the scope of this AP.

Resource constructs are interpreted to met the information requirements. This interpretation produces the application interpreted model (AIM). This interpretation, given in 5.1, shows the correspondence between the information requirements and the AIM. The short listing of the AIM specifies the interface to the integrated resources and is given in 5.2. Note that the definitions and *EXPRESS* provided in the the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. The expanded listing given in annex A contains the complete *EXPRESS* for the AIM without annotation. A graphical representation of the AIM is given in annex G. Additional requirements for specific implementation methods are given in annex D.

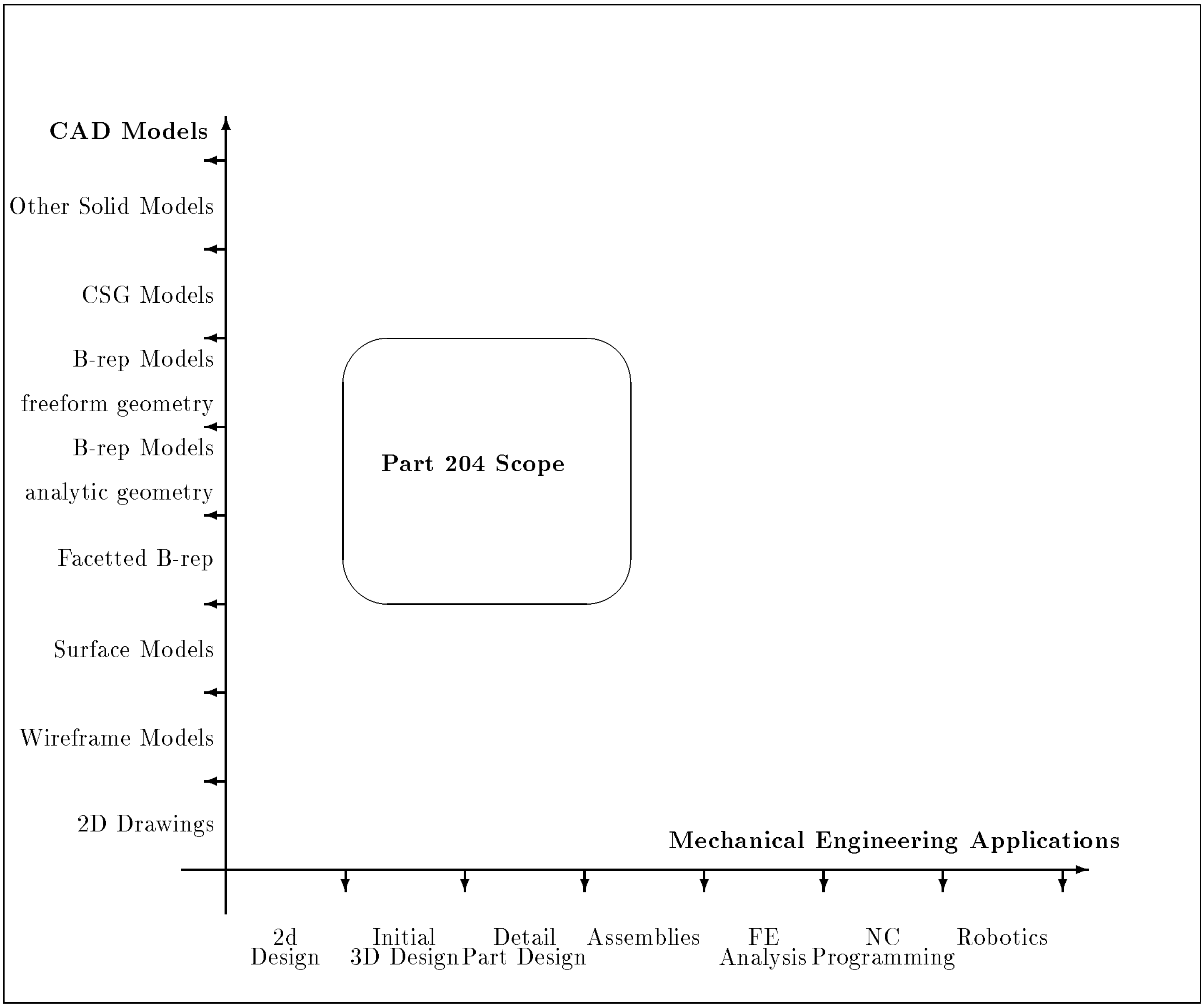

Figure 1 – The Scope of Part 204 in the contexts of CAD models and Mechanical Engineering Applications

Industrial automation systems and integration - Product data representation and exchange -

Part 204: Application protocol: Mechanical design using boundary representation

1 Scope

This part of ISO 10303 specifies the integrated resources necessary for the scope and information requirements for the use and exchange of boundary representation solid models in the mechanical engineering design context.

NOTE – The application activity model in annex E provides a graphical representation of the processes and information flows which are the basis for the definition of the scope of this part of ISO 10303.

This document describes an application reference environment for the generation and exchange of volume based design data in the Computer Aided Mechanical design process, together with appropriate data models and a physical file implementation form. The information model supports all geometric and topological aspects of a complete description of the shape and size of an object. It was originally developed for applications in mechanical engineering design using the CAD modelling technique boundary representation (B-rep) solid modelling and may be appropriate for other appliication areas using this technique.

The following are within the scope of this Part of ISO 10303:

– Three types of B-rep model that are used to represent shape:

a) facetted B-rep model;

b) B-rep model with elementary surfaces;

c) B-rep model with sculptured surfaces;

– curve and surface geometry;

– manifold topology;

– product identification information;

– the association of simple presentation attributes such as line-style, line-width, colour with a B-rep model or with geometric or topological elements;

– preservation of user-defined names of objects;

– units and measures associated with geometric elements;

– assemblies of parts and sub-assemblies.

The following are outside the scope of this Part of ISO 10303:

– Other types of shaperepresentation:

  a) wireframe models;

  b) surface models;

  c) geometrically trimmed curves and surfaces;

  d) constructive solid geometry models;

  e) compound B-rep models.

– Geometric and topological data:

  a) curves defined in parameter space (pcurves);

  b) 2D geometry;

  c) self-intersecting geometry;

  d) non-manifold topology.

– Dimensioning;

– Tolerances;

– Manufacturing information;

– Advanced presentation features, such as multiple views, character fonts, symbols;

– Material information;

– Meshing information;

– Analysis models, such as finite element analysis.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this this part of ISO 10303. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 10303-1:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 1 : Overview and fundamental principles.*

ISO 10303-11:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 11 : Description methods: The EXPRESS language reference manual.*

ISO 10303-21:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 21 : Implementation methods: Clear text encoding of the exchange structure.*

ISO/CD 10303-22[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 22 : Implementation methods: STEP data access interface specification.*

ISO 10303-31:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 31 : Conformance testing methodology and framework: General Concepts.*

ISO 10303-41:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 41 : Integrated resources: Fundamentals of product description and support.*

ISO 10303-42:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 42 : Integrated resources: Geometric and topological representation.*

ISO 10303-43:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 43 : Integrated resources: Representation structures.*

ISO 10303-46:–[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 46 : Integrated resources: Visual presentation.*

[1)] To be published.

ISO/CD 10303-1204[1)] *Industrial automation systems and integration - Product data representation and exchange - Part 1204 : Abstract test suite: Mechanical design using boundary representation.*

ISO TC 184/SC4 WG4 N603c[0)] *Application Interpreted Construct: Topology Bounded Surface*

ISO TC 184/SC4 WG4 N604c[−1)] *Application Interpreted Construct: Facetted Boundary Representation*

ISO TC 184/SC4 WG4 N605c[−2)] *Application Interpreted Construct: Elementary Boundary Representation*

ISO TC 184/SC4 WG4 N606c[−3)] *Application Interpreted Construct: Advanced Boundary Representation*

ISO TC 184/SC4 WG4 N618c[−4)] *Application Interpreted Construct: Mechanical Design Context*

ISO TC 184/SC4 WG4 N621[−5)] *Application Interpreted Construct: Mechanical Design Presentation*

3 Definitions

3.1 Terms defined in ISO 10303-1

This part of ISO 10303 makes use of the following terms defined in ISO 10303-1.

- **abstract test suite:**
- **application:**
- **application activity model (AAM):**
- **application context:**
- **application interpreted model (AIM):**
- **application object:**
- **application protocol (AP):**
- **application reference model (ARM):**
- **assembly:**
- **component:**
- **conformance class:**
- **conformance requirement:**
- **conformance testing:**
- **context:**
- **data:**
- **data exchange:**
- **implementation method:**
- **interpretation:**
- **integrated resource:**
- **model:**

- **presentation:**
- **product:**
- **product data:**
- **resource construct:**
- **structure:**
- **unit of functionality (UoF):**

3.2 Other definitions

For the purposes of this Part of ISO 10303, the following definitions apply.

3.2.1 advanced B-rep : a general boundary representation model which may have sculptured surface geometry for the faces.

3.2.2 bill-of-material structure: a structural description of a product in terms of its nested constituents.

3.2.3 Computer-aided Design: a way of designing mechanical and other products utilizing computerized tools.

3.2.4 component: a part which has a role as a constituent of an assembly.

3.2.5 constituent: a subdivision of a product, either a component or a sub-assembly.

3.2.6 elementary B-rep: a boundary representation model in which all surfaces are elementary.

3.2.7 facetted B-rep : a simplified boundary representation model with planar faces.

3.2.8 form, fit and function: the form is the shape of the product, the fit is the way the product interfaces with other products and the function is the purpose that the product serves in the overall product.

3.2.9 functional level: an indicator used to distinguish geometric complexity of a B-rep model or other representation.

3.2.10 genus: a topological property used to classify solids. The genus of a solid is an abstraction for the number of through holes in the solid.

3.2.11 normal direction : a unit vector perpendicular to a surface and pointing away from the material.

3.2.12 orientation: the mathematical sense of curves, loops or edges.

3.2.13 product version: an identifier for a variant of a product.

3.3 Abbreviations

For the purposes of this Part of ISO 10303, the following abbreviations apply.

B-rep: Boundary representation solid;

CAE: Computer Aided Engineering;

CIM: Computer Integrated Manufacturing;

CSG: Constructive Solid Geometry;

FEA: Finite Element Analysis;

ID: Identification;

IDEF0: ICAM definition language 0;

NIAM: Nijssen's Information Analysis Method;

NC: Numerical Control;

4 Information requirements

This clause specifies the information required for mechanical design using boundary representation.

The information requirements are specified as a set of units of functionality, application objects, and application assertions. These assertions pertain to individual application objects and to relationships between application objects. The information requirements are defined using the terminology of the subject area of this application protocol.

NOTES

1 – A graphical representation of the information requirements is given in annex F.

2 – The information requirements correspond to those of the activities identified as being in the scope of this application protocol in annex E.

3 – The mapping table is specified in 5.1 which shows how the information requirements are met using the integrated resources of this International Standard. The use of the integrated resources introduces additional requirements which are common to application protocols.

These requirements apply to system developers developing conforming implementations and to users of this application protocol to exchange physical files containing B-rep model data. An implementation claiming to conform to this application protocol shall ensure that the structure and constraints defined by these information requirements are satisfied when physical files are exchanged.

Functional Levels

The information requirements for mechanical design using boundary representation models are presented in terms of three distinct levels of functionality. The goal is to classify different implementations into levels distinguished by the complexity of the shape being represented.

The shape of each part described in this AP is composed of geometry and topology. The topology structure provides the connectivity and trimming information for the unbounded geometry of the part. In this Application Protocol the use of a topological entity requires that all associated geometry be defined. In order to classify different levels of design shape complexity the criterion used is complexity of surface geometry. Level 1 has simple surface geometry for each face of the model and much of the topological information is implicit. Both level 2 and level 3 provide for a complete explicit representation of the topology of the part in which all vertices, edges, loops and faces are included. The only distinction between level 2 and level 3 is in the complexity of the geometric curves and surfaces which are associated with the topological data. There is no distinction in topology structures between level 3 and level 2.

In this Part of ISO 10303 three levels of complexity are defined.

B-rep level 1: Level 1 geometric complexity is for models with planar surfaces as the bounding surfaces. Only points and planar polygons which can be implicitly represented by their vertex points are necessary for this representation.

At level 1 much of the topological information is implicit. Edge and vertex information is not given and the shells consist of faces bounded by polygons. The face geometry is implicitly planar. The complete part shape is represented by the polyhedron.

EXAMPLES

1 – box shapes;

2 – facetted shape approximating a model of more complex shape.

Level 1 models can either be an exact model of a simple part or a simplified model of a more complex part which is suitable for a selected range of applications such as stereolithography, or Finite Element Analysis.

Level 1 models can be represented in a more compact form than models from level 2 or level 3: edges and curves are not required to be explicitly defined since these are always straight lines. The connecting points are sufficient for their definition.

EXAMPLE 3 – Applications of these models:

a) in rapid prototype manufacturing;

b) for visualization purposes;

c) for collision checks of parts;

d) for kinematic studies;

e) for robot programming and simulations.

B-rep level 2: Level 2 of geometric complexity is for models with elementary surfaces. In this level the geometry needed to represent the curves and surfaces of objects is elementary analytic geometry. The surfaces included at this level are the plane, sphere, cylinder, cone, torus and swept surfaces. The curves are lines and conics. Both curves and surfaces are unbounded and the bounding information is contained in the topology data. At this level the complete part shape is represented by an elementary B-rep model.

EXAMPLES

4 – Application examples:

– milled parts suitable for $2\frac{1}{2}$D manufacturing;

– turned parts.

5 – Part examples:

– bolts and screws (excluding the thread detail);

– piston of a simple piston-engine;

– motor housings.

NOTES

4 – For the same part there can exist different model representations, which correspond to different application requirements.

5 – For a part there might exist a representation of level 1 and of level 2.

B-rep level 3: Level 3 of geometric complexity is for B-rep models with advanced surface descriptions.

This level will be used for modelling of parts whose geometric shape is representable with elementary , or sculptured surfaces, or swept surfaces with linear or rotational extrusions, or any combination of these. The generator curves for the extrusion can be analytic or free-form curves. The sculptured surfaces or free-form curves will be B-spline based. Level 3 includes more general forms of twisted curve and sculptured surface in addition to all those included in level 2.

EXAMPLES

6 – Application examples:

- parts which require 3 to 5 axis NC machining for their manufacturing;
- dies for moulding;
- dies for forming;
- ergonomically formed consumer products.

7 – Part examples:

- plastic housing of a telephone;
- car surface parts like fenders;
- housing block of a combustion engine.

NOTES

6 – Level 3 is a superset of level 2 in the sense that all entities supported at level 2 are also supported at level 3.

7 – Level 3 surfaces contain surfaces that may have any shape.

4.1 Units of functionality

This subclause specifies the units of functionality for the mechanical design using boundary representation application protocol. This Part of ISO 10303 specifies the following units of functionality:

- facetted_B-rep;
- elementary_B-rep;
- advanced_B-rep;
- name_preservation;
- product_structure;
- visual_presentation_for_B-rep.

The units of functionality and a description of the functions that each UoF supports are given below. The application objects included in the UoFs are defined in 4.2.

4.1.1 Facetted_B-rep

The facetted B-rep UoF provides the ability to define a boundary representation model all of whose faces are planar.

The following application objects are used by the facetted_B-rep UoF.

- B-rep;
- direction;
- face;
- facetted_B-rep;
- global_unit;
- loop;
- plane;
- point;
- poly_loop;
- shell;
- surface;
- void.

4.1.2 elementary_B-rep

The elementary_B-rep UoF provides for the definition of a boundary representation model composed of shells having topologically bounded elementary surfaces as faces.

The following application objects are used by the elementary_B-rep UoF.

- B-rep;
- bounded_curve;
- circle;
- closed_shell;
- conic;
- conical_surface;
- curve;
- cylindrical_surface;
- direction;
- edge;
- elementary_B-rep;
- elementary_surface;
- ellipse;
- face;
- global_unit;
- hyperbola;
- line;
- location;
- loop;
- parabola;
- plane;

- point;
- polyline;
- shell;
- spherical_surface;
- surface;
- toroidal_surface;
- unbounded_curve;
- vertex;
- void.

4.1.3 advanced_B-rep

The advanced_B-rep UoF provides the ability to define a boundary representation model with topologically bounded surfaces as faces and sculptured geometry.

The following application objects are used by the advanced_B-rep UoF.

- advanced_B-rep;
- B-rep;
- bounded_curve;
- circle;
- closed_shell;
- conic;
- conical_surface;
- curve;
- cylindrical_surface;
- direction;
- edge;
- elementary_surface;

- ellipse;
- face;
- global_unit;
- hyperbola;
- line;
- location;
- loop;
- parabola;
- plane;
- point;
- polyline;
- sculptured_surface;
- shell;
- spherical_surface;
- surface;
- surface_of_extrusion;
- surface_of_revolution;
- swept_surface;
- toroidal_surface;
- twisted_curve;
- unbounded_curve;
- vertex;
- void.

4.1.4 name_preservation

The name_preservation UoF provides the ability to associate and preserve user defined names with models or model components. This allows for the preservation of entity names that were defined by a user by means of a computer aided application. The user-defined name of an item is used as an alias to any implementation dependent identifiers.

The following application objects are used by the name_preservation UoF.

- name.

4.1.5 product structure

The product_structure UoF provides the ability to define a product as an assembly of parts or of sub-assemblies. In this AP each part is defined as a B-rep model. Products are composed of individual parts and of collections of parts which form so called assemblies. Assemblies may consist of sub-assemblies and of individual parts. Individual parts are represented by specific geometric shape descriptions as B-rep models. Assemblies have specific geometric relationships with one another and to individual parts. This UoF includes the structures for the identification of mechanical parts assemblies and the structure that links the shape of the parts and assemblies to their identification.

These relationships are given by the following properties:

- a reference from an assembly to another assembly or to a part;

- a geometric relationship, which may be described with a transformation matrix, which allows translation, rotation, and if required mirroring and scaling;

- the assemblies and parts have names which may be required to be unique within one product assembly.

The following detailed requirements are met by this UoF:

- the structural description of assemblies and parts with references to shapes;

- the assembly structure shall be independent of the referenced shape of the product;

- the versioning of parts;

- the identification of multiple definitions of parts;

- the specification of the relationship between parts which comprise an assembly;

- the specification of the relationship between a part and its shape representation;

- the specification of the relationship of a part shape to the shape representation of an assembly.

The following application objects are used by the product_structure UoF.

- assembly;
- part;
- product;
- shape_representation;
- transformation.

4.1.6 visual_presentation_for_B-rep

The visual_presentation_for_B-rep UoF consists of application objects for the association of elementary viewing information with geometric or topological components of B-rep models. This enables the user to to specify the visual appearance for points, curves and surfaces and to assign pre-defined colour, line-style (dashed, dotted and similar) and line-width to any geometric or topological entity. Annotation text can be positioned in the 2D-space of a screen image. A layer mechanism is provided for display purposes. Layers may contain geometric and topological entities as well as complete geometric models. An entity may belong to several layers.

The following application objects are used by the visual_presentation_for_B-rep UoF.

- 3D_projection;
- annotation_text;
- curve_appearance;
- layer;
- point_appearance;
- presentation_appearance;
- screen_image;
- surface_appearance.

4.1.7 Relationship of units of functionality to functional levels

NOTE – Table 1 shows the relationship between the units of functionality in this AP and the functional levels used in the definition of conformance classes.

4.2 Application Objects

This subclause specifies the application objects for the mechanical design using boundary representation application protocol. Each application object is an atomic element which embodies

B-rep level 1	B-rep level 2	B-rep level 3
name preservation		
product structure		
visual presentation for B-rep		
facetted B-rep	elementary B-rep	advanced B-rep

Table 1 – Use of units of functionality within functional levels

a unique application concept and contains attributes specifying the data elements of the object. The application objects and their definitions are given below.

4.2.1 3D_Projection

A 3D_projection is a 2-dimensional picture of a 3-dimensional shape. The picture is the image of a mapping defined by a camera model.

4.2.2 Advanced_B-rep

The advanced B-rep is a type of B-rep (see 4.2.4) which may include elementary geometry, sculptured geometry and swept geometry.

4.2.3 Assembly

An Assembly is a collection of assemblies or parts which are connected together. It has a name assigned, which may describe the functionality, and uses transformations to position its sub-assemblies or components in space.

The data associated with an Assembly are the following:

- coordinate_system;
- user_defined_name.

4.2.3.1 coordinate_system

The coordinate_system identifies the Cartesian coordinate system used to define the geometry of the Assembly. This is the underlying global rectangular Cartesian coordinate system to which all geometry refers. A coordinate_system is identified with the context of the shape representation for an Assembly.

4.2.3.2 user_defined_name

The user_defined_name is a user defined identifier for the Assembly. It consists of one or more words and may describe the functionality of the Assembly.

4.2.4 B-rep

A B-rep is a manifold boundary representation which defines a volume in terms of topology and geometry. The geometry is used to describe the shape of the object in terms of surfaces and edge-curves. The topology bounds the geometry to a finite extent and combines all bounded elements to define a manifold solid. The material side of the solid is defined by normal vectors to the faces, this normal shall point away from the material. A B-rep may be either an Advanced_B-rep (see 4.2.2), an Elementary_B-rep (see 4.2.15), or a Facetted_B-rep (see 4.2.19).

NOTE – see figure F.3 in NIAM diagrams.

4.2.5 Bounded_curve

A Bounded_curve is a type of curve (see 4.2.10) with finite extent and identifiable end points. A Bounded_curve may be a Polyline (see 4.2.33), or a Twisted_curve (see 4.2.50)

4.2.6 Circle

A Circle is a type of Conic (see 4.2.8) defined by its radius, centre point and the normal direction of the intersection plane.

4.2.7 Closed_shell

A Closed_shell is a collection of one or more Faces which bounds a region in 3-dimensional space. The topological normal of the shell is defined as being directed from the finite to the infinite region.

4.2.8 Conic

A Conic is a planar curve which may be produced as the intersection of a plane and a conical surface, where the axis of the conic does not lie in the plane. Each Conic is a type of Curve (see 4.2.10) which is either a Circle (see 4.2.6), Ellipse (see 4.2.17), Hyperbola (see 4.2.22), or Parabola (see 4.2.28).

4.2.9 Conical_surface

A Conical_surface is a type of Elementary_surface (see 4.2.16) constructed by moving a line, which is fixed in one point, the vertex point, along a closed circle.

4.2.10 Curve

A Curve is a one dimensional manifold in a space of dimension 2 or 3. Informally, a Curve can be envisioned as the path of a point moving in its coordinate space. A Curve may be either a Line (see 4.2.24), a Conic (see 4.2.8), or a Bounded_curve (see 4.2.5).

NOTE – see figure F.9 in the NIAM diagrams.

4.2.11 Curve_appearance

Curve_appearance specifies the required appearance of a curve when it is visualised.

The data associated with a Curve_appearance are the following:

- colour;
- curve_font;
- curve_width.

4.2.11.1 colour

The colour defines the colour of the Curve when viewed with a colour display. It is specified by intensity values for red, green and blue. The colour need not be specified for a particular Curve in which case the default option is black.

4.2.11.2 curve_font

The curve_font describes the lengths of the visible and invisible segments of a curve when visualised. The curve_font need not be specified for a particular Curve in which case the default option is fully visible.

4.2.11.3 curve_width

The curve_width specifies the apparent width of the curve when visualised. The curve_width need not be specified for a particular Curve in which case the default option is that of the graphics system.

4.2.12 Cylindrical_surface

A Cylindrical_surface is a type of Elementary_surface (see 4.2.16) constructed by the parallel shifting of a line along a closed circle, where the line is parallel to the normal to the plane of the circle. A Cylindrical_surface is defined by its radius, axis direction and a point on the axis.

4.2.13 Direction

A Direction is a unit vector in 3 dimensional space.

4.2.14 Edge

An Edge is the bounded intersection curve of two surfaces. It is defined by the curve geometry and two vertices.

4.2.15 Elementary_B-rep

The Elementary_B-rep is a type of B-rep (see 4.2.4) which consists of only elementary geometric elements, all Faces and Edges being associated with elementary geometry.

4.2.16 Elementary_surface

An Elementary_surface is a type of Surface (see 4.2.42) which can be described by simple analytic equations. An Elementary_surface may be a Plane (see 4.2.30), a Spherical_surface (see 4.2.41), a Conical_surface (see 4.2.9), Cylindrical_surface (see 4.2.12), or a Toroidal_surface (see 4.2.48).

NOTE – see figure F.8 in the NIAM diagrams

4.2.17 Ellipse

An ellipse is a type of Conic (see 4.2.8) defined by the lengths of the semi-major and semi-minor diameters, centre point and the normal direction to its plane.

4.2.18 Face

A Face is a bounded portion of a Surface (see 4.2.42). It consists of the surface geometry, at least one surrounding loop and possible inner loops which can be regarded as holes in the surface. If the Face is used in a Facetted_B-rep (see 4.2.19) the loops are simplified and defined as Poly_loops (see 4.2.34).

4.2.19 Facetted_B-rep

A Facetted_B-rep is a type of B-rep (see 4.2.4) with only planar surfaces and straight lines as geometric elements. The topology is simplified by using Poly_loops (see 4.2.34) which list all points defining the corners of the faces of the solid.

4.2.20 Geometric_element

A Geometric_element is part of the geometric description of a B-rep. A Geometric_element is either a Point, a Curve (see 4.2.10), or a Surface (see 4.2.42).

4.2.21 Global_unit

A unit that is valid for all the corresponding types of measure within the context of a shape model.

EXAMPLE 8 – millimetre could be a global unit for length measures.

4.2.22 Hyperbola

A Hyperbola is a type of Conic (see 4.2.8) which is an open conic section defined by its radius, imaginary radius, the centre point and the normal direction to its plane.

4.2.23 Layer

A Layer is a grouping mechanism, which may be recursive, for the visualization of one or more B-rep models, topological entities and geometric entities. A Layer may be assigned a name. The data associated with a layer is the following:

- user_defined_name.

The user_defined_name specifies one or more words chosen by the user to identify the layer. The user_defined_name need not be specified for a particular Layer.

4.2.24 Line

A line is a type of Curve (see 4.2.10) which is unbounded and is defined by a point and a direction. The positive direction of the Line is that of the direction vector.

4.2.25 Location

A Location is the placement, or position and orientation, of a geometric element in the coordinate space. It is defined by a Point and 2 Directions (see 4.2.31 and 4.2.13).

4.2.26 Loop

A Loop is the boundary of a surface. The Loop has to be closed and self-intersection is not allowed. A Loop consists of an ordered collection of at least one Edge (see 4.2.14) in the case of edge loop, or of a single Vertex (see 4.2.52) in the case of vertex_loop, or of an ordered collection of points in the case of a Poly_loop (see 4.2.34).

4.2.27 Name

A Name is a user defined identifier for an object. Any entity of this Part that may be part of a B-rep model can be assigned a Name. In addition layers may have such Names. Presentation attributes shall not have Names.

4.2.28 Parabola

A Parabola is a type of Conic (see 4.2.8), and is defined by its focal length, i.e., the distance between focal point and vertex point, vertex point and the direction of the normal to its plane.

4.2.29 Part

A Part is a mechanical component which can be represented by a B-rep solid model. The data associated with a part is the following:

- user_defined_name.

The user_defined_name specifies one or more words chosen by the user to identify and describe the functionality of the part.

4.2.30 Plane

A plane A Plane is a type of Surface (see 4.2.42) which is unbounded and has a constant normal. A Plane is defined by a point on the surface and a normal direction.

4.2.31 Point

A Point is a precise location in real space. A point is defined by its Cartesian coordinates in 3-dimensional space.

4.2.32 Point_appearance

A Point_appearance specifies the visual appearance of a Point. (see 4.2.31)

The data associated with a Point_appearance are the following:

- colour;
- marker;
- marker_size.

4.2.32.1 colour

The colour specifies the colour of the Point when viewed with a colour display. It is specified by intensity values for red, green and blue. The colour need not be specified for a particular Point in which case the default option is black.

4.2.32.2 marker

The marker specifies the form of the symbol used to display a Point. The marker need not be specified for a particular Point in which case the default option is an addition sign.

4.2.32.3 marker_size

The marker_size specifies the size of the selected marker type. The marker_size need not be specified for a particular Point in which case the default value for this size is 1mm.

4.2.33 Polyline

A polyline is a type of Bounded_curve (see 4.2.5) which consists of n-1 linear segments. It is defined by a list of n points.

4.2.34 Poly_loop

A Poly_loop is a type of Loop (see 4.2.26) used in the Facetted_B-rep (see 4.2.19). It is represented by an ordered coplanar collection of points forming the vertices of the loop. The loop is composed of straight line segments each joining a point in the collection to the succeeding point in the collection. The closing segment is from the last to the first point in the collection. The direction of the loop is in the direction of the line segments.

4.2.35 Presentation_appearance

A Presentation_appearance is a specification of the visual appearance which is relevant for the visualization of geometric models. The visual properties which may be specified include curve-style and curve-width for curves, colour for B-rep models, surfaces, curves, points and 3D-annotation text. Presentation_appearance is an abstract concept which will not itself be instantiated but may be assigned to geometric elements, topological elements, or to layers.

4.2.36 Product

A product is a physical manufactured object. In the context of this Part of ISO 10303 the shape of a product is represented by one or more B-rep models corresponding to the constituent parts of the Product. These may be collected together as an Assembly (see 4.2.3).

The data associated with a product are the following:

- coordinate_system;
- user_defined_name;
- version_and_id.

4.2.36.1 coordinate_system

The coordinate_system is the Cartesian coordinate system used to define the precise geometry of the Product.

4.2.36.2 user_defined_name

The user_defined_name is the name selected by the user for reference purposes. It consists of one or more words and may describe the product functionality.

4.2.36.3 version_and_id

The version_and_id specifies terms including version number and identifier which uniquely identify an instance of a Product.

4.2.37 Sculptured_surface

A Sculptured_surface is a type of Surface (see 4.2.42), which is a general bi-parametric surface of polynomial or rational form.

4.2.38 Screen_image

A Screen_image is a collection of 2-dimensional images and text which is intended to be displayed simultaneously. There is a maximum of one Screen_image present in any instance of a model which conforms to this Part of ISO 10303.

4.2.39 Shape_representation

A Shape_representation is a representation of the precise size and shape of a mechanical product or component. A Shape_representation is made up of one or more B-rep models (see 4.2.4).

4.2.40 Shell

A in the context of this AP is a closed shell and is a collection of one or more Faces (see 4.2.18) which bounds a region in the 3-dimensional space. The topological normal of the Shell is defined as being directed from the finite to the infinite region. A Void (see 4.2.53) in a B-rep is represented by an interior shell which builds a "hole" inside an outer shell. For an interior shell the topological normal will point into the solid material.

NOTE – see figure F.6 in the NIAM diagrams

.

4.2.41 Spherical_surface

A Spherical_surface is a ball shaped surface. It is a type of Elementary_surface (see 4.2.16) and is defined by its radius and its centre point.

4.2.42 Surface

A surface can be regarded as being generated by a continuously changing curve moving in space. A Surface is either an Elementary_surface (see 4.2.16), a Swept_surface (see 4.2.46), or a Sculptured_surface (see 4.2.37). The extent of a surface may be infinite.

NOTE – see figure F.7 in the NIAM diagrams.

4.2.43 Surface_appearance

A Surface_appearance is a specification of the visual appearance of a Surface (see 4.2.42) when displayed.

The data associated with a Surface_appearance are the following:

- colour;
- grid_indicator;
- shading_method.

4.2.43.1 colour

The colour specifies the colour of the Surface when viewed with a colour display. It is specified by intensity values for red, green and blue. The colour need not be specified for a particular Surface in which case the default option is black.

4.2.43.2 grid_indicator

The grid_indicator describes the way in which a surface is to be displayed. This includes the selection of the curves which are used to display the Surface. The Surface may be presented using one or more of the following options:

- boundary curves;
- silhouette curves;
- segmentation curves;
- control grid;
- parameter lines;
- shading.

The grid_indicator need not be specified for a particular Surface in which case the default option is silhouette curves.

4.2.43.3 shading_method

The shading_method defines the algorithm used for the shading of a surface. The shading_method need not be specified for a particular Surface in which case the default option is normal shading, this is based upon interpolated values of the surface normals.

4.2.44 Surface_of_extrusion

A Surface_of_extrusion is a simple surface constructed by sweeping a Curve (see 4.2.10) in a given fixed direction. A Surface_of_extrusion is a type of Swept_surface (see 4.2.46).

4.2.45 Surface_of_revolution

A Surface_of_revolution is the surface constructed by rotating a Curve (see 4.2.10) a complete revolution about an axis. A Surface_of_revolution is a type of Swept_surface (see 4.2.46).

4.2.46 Swept_surface

A Swept_surface is one that is constructed by sweeping a curve along another curve. Each Swept_surface is either a Surface_of_extrusion (see 4.2.44) or a surface_of_revolution (see 4.2.45).

4.2.47 Topological_element

A Topological_element is part of the topological definition of a B-rep. A Topological_element is either a Vertex, an Edge (see 4.2.14), a Loop (see 4.2.26), a Face (see 4.2.18), or a Shell (see 4.2.40).

4.2.48 Toroidal_surface

A Toroidal_surface is a surface constructed by rotating a circle around an axis lying outside the circle and in the plane of the circle. It is defined by a centre point, an axis and by two radii. A Toroidal_surface is a type of Elementary_surface (see 4.2.16).

4.2.49 Transformation

A Transformation is a geometric operation composed of translation, rotation, mirroring and uniform scaling. Transformations may be used to locate the components of an assembly.

4.2.50 Twisted_curve

A Twisted_curve, in the context of this AP, is a general parametric 3-dimensional curve which is represented by a B-spline curve. A Twisted_curve is a type of Bounded_curve (see 4.2.5).

4.2.51 Unbounded_curve

An Unbounded_curve is a curve with infinite extent. An Unbounded_curve is either a Line (see 4.2.24), or a Conic (see 4.2.8).

4.2.52 Vertex

A Vertex is a topological point which forms part of the boundary of an Edge or a Face.

4.2.53 Void

A Void is a hollow region inside a solid model. In this AP a void is represented by a Shell with a normal pointing into the empty space inside, this reverses the sense of the topological normal to the defining shell.

4.3 Application assertions

This subclause specifies the application assertions for the mechanical design using boundary representation application protocol. The application assertions define the required relationships between the application objects defined in the previous clause, the cardinality of the relationships and any integrity rules required.

4.3.1 3D_projection to Screen_image

A 3D_projection is in one Screen_image. A Screen_image contains one, or many 3D_projections.

4.3.2 3D_projection to B-rep

A 3D_projection presents zero, one or many B-rep models. A B-rep model is associated with zero or one or many 3D_projections.

4.3.3 Annotation_text to Screen_image

An Annotation_text is presented in one Screen_image. A Screen_image may contain zero, one, or many Annotation_texts.

4.3.4 Annotation_text to Transformation

An Annotation_text is located by precisely one Transformation. Each Transformation may be used to locate zero, one or many Annotation_texts.

4.3.5 Assembly to Assembly

An Assembly may contain one or more Assemblies, in the role of sub-assemblies with associated transformation. An Assembly is part of zero, one or many Assemblies.

4.3.6 Assembly to Product

An Assembly is part of one or more Products. A Product may consist of zero, one or more Assemblies.

4.3.7 B-rep to Closed_shell

A B-rep shall consist of one or more Closed_shells. Where two or more Closed_shells are used to define a B-rep one of these shall be the outer shell and contain all the others.

4.3.8 Closed_shell to Face

A Closed_shell is defined by one or more Faces. A Face shall be used to define one or more Closed_shells. Within a given B-rep model a Face shall be associated with precisely one Closed_-shell.

4.3.9 Curve to Edge

Each Curve in the exchange structure shall be used to define the geometry of one or more Edges.

4.3.10 Curve_appearance to Curve

A curve_appearance may be associated with zero, one, or many Curves. A Curve is represented as zero or one Curve_appearances.

4.3.11 Curve_appearance to Edge

A curve_appearance may be associated with zero, one, or many Edges. An Edge is represented as zero or one curve_appearances.

4.3.12 Edge to B-rep

Each Edge referenced by a B-rep model shall be used precisely twice with opposing orientations in the definition of that B-rep model.

4.3.13 Edge to Curve

Each Edge shall have its geometry defined by precisely one Curve.

4.3.14 Edge to Edge

An Edge shall not intersect any other Edge except at a shared Vertex.

4.3.15 Edge to Vertex

An Edge is defined by precisely two vertices.

4.3.16 Face to Face

A face shall not intersect any other face except along a common edge.

4.3.17 Face to Loop

A Face shall reference one or more Loops.

4.3.18 Face to Surface

A Face shall have its geometry defined by precisely one Surface. In a facetted B-rep model the geometry of each Face shall be defined by precisely one Plane. Each Surface in the exchange structure shall be used to define the geometry of one or more Faces.

4.3.19 Layer to Assembly

A Layer contains zero one or many Assemblies. An Assembly may appear in zero, one or many Layers.

4.3.20 Layer to B-rep

A Layer contains zero one or many B-rep models. A B-rep model may appear in zero, one or many Layers.

4.3.21 Layer to Geometric_element

A Layer contains zero, one or many Geometric_elements. A Geometric_element may be associated with zero, one, or more, Layers for the purposes of visual presentation.

4.3.22 Layer to Layer

A Layer contains zero one or many Layers. A Layer may be assigned to zero, one or many Layers provided there is no recursion.

4.3.23 Layer to Part

A Layer contains zero one or many Parts. A Part may be assigned to zero, one or many Layers.

4.3.24 Layer to Presentation_appearance

A layer is presented by zero, one, or many presentation_appearances. A presentation_appearance presents zero, one, or many layers.

4.3.25 Layer to Product

A layer contains zero one or many products. A product may be assigned to zero, one or many layers.

4.3.26 Layer to Topological_element

A Layer contains zero, one or many Topological_elements. A topological_element may be associated with zero, one, or more, Layers for the purposes of visual presentation.

4.3.27 Loop to Edge

A Loop of type edge_loop shall be defined by one or Edges. Each Edge shall be used in the definition of one or two Loops in a B-rep model.

4.3.28 Name to Geometric_element

A Geometric_element may have zero or one user defined Name.

4.3.29 Name to Topological_element

A Topological_element may have zero or one user defined Name.

4.3.30 Part to assembly

A Part may belong to one or more Assemblies. An Assembly contains one or more Parts. Each Part in the Assembly is located by one Transformation. An Assembly may not exist without a Part.

4.3.31 Part to Product

A Part is part of zero, one or many Products. A Product consists of zero, one, or many Parts.

4.3.32 Part to Shape_representation

A Part shall have its shape defined by one or more B-rep models in a Shape_representation.

4.3.33 Point_appearance to Point

A Point_appearance may be associated with zero, one, or many Points. A Point is represented as zero or one Point_appearances.

4.3.34 Presentation_appearance to B-rep

A Presentation_appearance presents zero, one, or many B-reps. A B-rep model is presented by zero, one or many Presentation_appearances.

4.3.35 Presentation_appearance to Shell

A Presentation_appearance presents zero, one, or many Shells. A Shell is presented by zero, one or many Presentation_appearances.

4.3.36 Shape_representation to B-rep

Each B-rep model in the exchange structure shall be part of precisely one Shape_representation corresponding to one of the three levels of functionality defined in this document. The simplest possible form of representation compatible with the information requirements shall be used.

4.3.37 Surface_appearance to Face

A Surface_appearance presents zero, one, or many Faces. A Face is presented by zero or one Surface_appearances.

4.3.38 Surface_appearance to Surface

A Surface_appearance presents zero, one, or many Surfaces. A Surface is presented by zero or one Surface_appearances.

4.3.39 Topology to Presentation_attribute

An item of topology may be associated with zero, one, or more, presentation attributes to define its display properties.

4.3.40 Transformation to Assembly

A Transformation may be used to locate an Assembly, as a sub-assembly, in an Assembly. When an assembly has the role of a sub-assembly it shall be located by a Transformation.

4.3.41 Transformation to Part

A Transformation may be used to locate a Part in an Assembly. Each Part in an Assembly shall be located by a Transformation.

4.3.42 Vertex to Point

Each Vertex shall have its geometry defined by precisely one Point. A Point may be used to define the geometry of zero or one Vertex.

5 Application interpreted model

5.1 Mapping table

This clause contains the mapping table that shows how each UoF and application object of this part of ISO 10303 (see clause 4) maps to one or several resource constructs (see 5.2).

The mapping table is organized in five columns. The contents of these five columns are:

– Column 1) Application element: Name of an application element as it appears in the application object definition in 4.2. Application object names are written in uppercase. Attribute names and assertions are listed after the application object to which they belong and are written in lower case.

– Column 2) AIM element: Name of an AIM element as it appears in the AIM (see 5.2). AIM entities are written in lower case. Attribute names of AIM entities are referred to as <entity name> . <attribute name>. The mapping of an application element may result in several related AIM elements. Each of these AIM elements requires a line of its own in the table.

– Column 3) Source: For those AIM elements that are interpreted from the integrated resources, this is the number of the corresponding part of ISO 10303. For those AIM elements that are created for the purpose of this part of ISO 10303, this is the number of the part being written. For those AIM elements that are directly incorporated from an application interpreted construct (AIC) this is the AIC reference.

– Column 4) Rules: One or more numbers may be given that refer to rules that apply to the current AIM element. For rules that are derived from relationships between application objects, the same rule is referred to by the mapping entries of all the involved AIM elements. Global rules are given for each application object or AIM element to which they apply. Immediately following the table, the rules are listed by number and by expanded name.

– Column 5) Reference path: To describe fully the mapping of an ARM element it may be necessary to specify a reference path through several related AIM elements. A single AIM element is documented on a single row within the reference path column with a symbol that defines its relationship to the AIM element on the succeeding row in the column. The reference path column, therefore, documents the role of an AIM element relative to the AIM element in the row suceeding it. Two or more such related AIM elements define the interpretation of the integrated resources that satisfies the requirement specified by the ARM element if a reference path is provided.

For each AIM element that has been created for use within this part of ISO 10303, a reference path up to its supertype from an integrated resource or an AIC is specified. In the case of a bi-directional reference from an AIM element with two attributes, each of which spawns a reference path, each reference path is enclosed by a set of parentheses. The AIM element that is the root of both reference paths is documented either between the sets of parentheses or preceeding each set of parentheses.

For the expression of reference paths the following notational conventions apply:

a) − > : attribute references the entity or select type in the following row;

b) < − : attribute is entity or select type referenced by the attribute in the following row;

c) => : entity is a supertype of the entity in the following row;

d) <= : entity is a subtype of the entity in the following row;

e) = : attribute has as its type the specified select or enumeration type, possibly constrained to particular choices or values.

This Part of ISO 10303 makes use of the following application interpreted constructs (AIC):

– AIC1 = aic_mech_dsgn_ctxt : mechanical design context AIC;

– AIC2 = aic_top_bnd_surf : topology bounded elementary surface AIC;

– AIC3 = aic_fac_brep : facetted B-rep AIC;

– AIC4 = aic_el_brep : elementary B-rep AIC;

– AIC5 = aic_adv_brep : advanced B-rep AIC (uses AIC2);

– AIC6 = aic_mech_dsgn_pres : mechanical design presentation AIC.

NOTE – The geometric AICs AIC2 to AIC5 are interdependent and their relationships via 'USE FROM' statements are illusrated in figure 2.

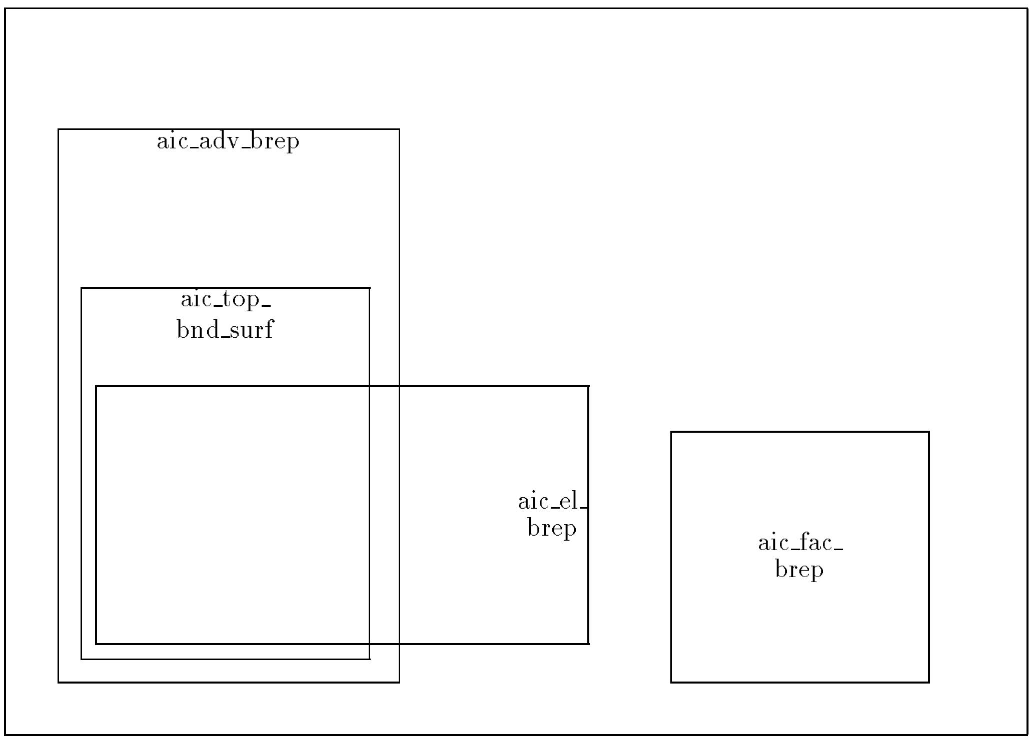

Figure 2 – Relationships between geometric AICs

Table 2 – ARM to AIM mapping table for Advanced B-rep UoF

Application element	AIM element	Source	Rules	Reference Path
ADVANCED_B-REP	advanced_brep_shape_representation	AIC5 41	1	advanced_brep_shape_representation ⇐ shape_representation
B-REP	manifold_solid_brep OR brep_with_voids	AIC5 AIC5	2,9,12	
B-rep to closed_shell				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell)
BOUNDED_CURVE	bounded_curve	AIC2	9,12	
CIRCLE	circle	AIC2	2,9,12	
CLOSED_SHELL	closed_shell	AIC5	7	
closed_shell to face				closed_shell ⇐ connected_face_set connected_face_set.cfs_faces → face
CONIC	conic	AIC2	2,9,12	
CONICAL_SURFACE	conical_surface	AIC4	2,9,12	
CURVE	curve	AIC2	9,12	
curve to edge				curve ← edge_curve.edge_geometry edge_curve
CYLINDRICAL_SURFACE	cylindrical_surface	AIC4	2,9,12	
DIRECTION	direction	AIC2	2,9,12	
EDGE	edge_curve	AIC2 42	7,9,12	edge_curve ⇐ edge

ARM to AIM mapping table for Advanced B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
edge to B-rep				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell) closed_shell.cfs_faces[i] → face face.bounds[i] → face_bound face_bound.bound → loop ⇒ edge_loop edge_loop\path.edge_list[i] → oriented_edge oriented_edge.edge_element → edge
edge to curve				edge_curve.edge_geometry → curve
edge to edge				edge_curve ⇐ edge (edge.edge_start) (edge.edge_end) → { vertex_point = vertex_point } ← (edge.edge_start) (edge.edge_end) edge
edge to vertex				(edge.edge_start)(edge.edge_end) → vertex_point
ELEMENTARY_SURFACE	elementary_surface	AIC2	9,12	
ELLIPSE	ellipse	AIC2	2,9,12	
FACE	face_surface	AIC2	2,9,12	

ARM to AIM mapping table for Advanced B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
face to face face to loop				face.bounds[i] → face_bound face_bound.bound → loop ⇒ edge_loop edge_loop\path.edge_list[i] → oriented_edge oriented_edge.edge_element → {edge = edge} ← oriented_edge.edge_element oriented_edge ← edge_loop\path.edge_list[i] edge_loop ⇐ loop ← face_bound.bound face_bound ← face.bounds[i] face_surface ⇐ face face.bounds[i] → face_bound face_bound.bound → loop ⇒
face to surface				face_surface face_surface.face_geometry → surface ⇒
GLOBAL_UNIT	global_unit_assigned_context.units	41	8	
HYPERBOLA	hyperbola	AIC2	2,9,12	
LINE	line	AIC2	2,9,12	
LOCATION	axis2_placement_3d	AIC2	2	
LOOP	loop	AIC2	7	

ARM to AIM mapping table for Advanced B-rep UoF (concluded)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
loop to edge				edge_loop ⇐ path path.edge_list[i] → oriented_edge ⇐ edge
PARABOLA	parabola	AIC2	2,9,12	
PLANE	plane	AIC2	2,9,12	
POINT	cartesian_point	AIC2	2,9,12	
POLYLINE	polyline	AIC2	2,9,12	
SCULPTURED_SURFACE	b_spline_surface	AIC2	2,9,12	
SHELL	closed_shell	AIC5	7	
SPHERICAL_SURFACE	spherical_surface	AIC2	2,9,12	
SURFACE	surface	AIC2		
SURFACE_OF_EXTRUSION	surface_of_linear_extrusion	AIC2	2,9,12	
SURFACE_OF_REVOLUTION	surface_of_revolution	AIC2	2,9,12	
SWEPT_SURFACE	swept_surface	AIC2	9	
TOROIDAL_SURFACE	toroidal_surface	AIC2	2,9,12	
TWISTED_CURVE	b_spline_curve	AIC2	2,9,12	
UNBOUNDED_CURVE	conic OR line	AIC2 AIC2		
VERTEX	vertex_point	AIC2	2,7,12	vertex_point
vertex to point				vertex_point.vertex_geometry → cartesian_point
VOID	brep_with_voids.voids[i]	AIC5		

Table 3 – ARM to AIM mapping table for Elementary B-Rep UoF

Application element	AIM element	Source	Rules	Reference Path
B-REP	manifold_solid_brep OR brep_with_voids	AIC4 AIC4	2,9,12	
B-rep to closed_shell				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell)
BOUNDED_CURVE	polyline	AIC4	2,9,12	
CIRCLE	circle	AIC4	2,9,12	
CLOSED_SHELL	closed_shell	AIC5	7	
closed_shell to face				closed_shell ⇐ connected_face_set connected_face_set.cfs_faces → face
CONIC	conic	AIC4	2,9,12	
CONICAL_SURFACE	conical_surface	AIC4	2,9,12	
CURVE	curve	AIC4	9,12	
curve to edge				curve ← edge_curve.edge_geometry edge_curve
CYLINDRICAL_SURFACE	cylindrical_surface	AIC4	2,9,12	
DIRECTION	direction	AIC4	2,9,12	
EDGE	edge_curve	AIC4 42	7,9,12	edge_curve ⇐ edge
edge to B-rep				(manifold_solid_brep.outer → closed_shell) (brep_with_voids.voids[i] → oriented_closed_shell ⇐ closed_shell)

ARM to AIM mapping table for Elementary B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
				closed_shell.cfs_faces[i] → face face.bounds[i] → face_bound face_bound.bound → loop ⇒ edge_loop edge_loop\path.edge_list[i] → oriented_edge oriented_edge.edge_element → edge
edge to curve				edge_curve.edge_geometry → curve
edge to edge				edge_curve ⇐ edge (edge.edge_start) (edge.edge_end) → {vertex_point = vertex_point } ← (edge.edge_start) (edge.edge_end) edge
edge to vertex				(edge.edge_start)(edge.edge_end) → vertex_point
ELEMENTARY_B-REP	elementary_brep_shape_ representation	AIC4 41	1	elementary_brep_shape_representation ⇐ shape_representation
ELEMENTARY_SURFACE	elementary_surface	AIC4	9	
ELLIPSE	ellipse	AIC4	2,9,12	
FACE	face_surface	AIC4	7,9,12	
face to face				face.bounds[i] → face_bound face_bound.bound →

ARM to AIM mapping table for Elementary B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
				loop ⇒ edge_loop edge_loop\path.edge_list[i] → oriented_edge oriented_edge.edge_element → {edge = edge} ← oriented_edge.edge_element oriented_edge ← edge_loop\path.edge_list[i] edge_loop ⇐ loop ← face_bound.bound face_bound ← face.bounds[i]
face to loop				face_surface ⇐ face face.bounds[i] → face_bound face_bound.bound → loop ⇒
face to surface				face_surface face_surface.face_geometry → surface ⇒ elementary_surface
GLOBAL_UNIT	global_unit_assigned_context.units	41	8	
HYPERBOLA	hyperbola	AIC4	2,9,12	

ARM to AIM mapping table for Elementary B-rep UoF (concluded)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
LINE	line	AIC4	2,9,12	
LOCATION	axis2_placement_3d	AIC4	2,9	
LOOP	loop	AIC4	7	
loop to edge				edge_loop ⇐ path path.edge_list[i] → oriented_edge ⇐ edge
PARABOLA	parabola	AIC4	2,9,12	
PLANE	plane	AIC4	2,9,12	
POINT	cartesian_point	AIC4	2,9,12	
POLYLINE	polyline	AIC4	2,9,12	
SHELL	closed_shell	AIC4	7	
SPHERICAL_SURFACE	spherical_surface	AIC4	2,9,12	
SURFACE	elementary_surface	AIC4	9,12	
TOROIDAL_SURFACE	toroidal_surface	AIC4	2,9,12	
UNBOUNDED_CURVE	conic OR line	AIC2 AIC2		
VERTEX	vertex_point	AIC4	7,9,12	vertex_point
vertex to point				vertex_point.vertex_geometry → cartesian_point
VOID	brep_with_voids.voids[i]	AIC4		

Table 4 – ARM to AIM mapping table for Facetted B-Rep UoF

Application element	AIM element	Source	Rules	Reference Path
B-REP	facetted_brep OR (facetted_brep AND brep_with_voids)	AIC3 AIC3	2,9,12	
B-rep to closed_shell	manifold_solid_brep.outer OR brep_with_voids.voids[i]	AIC3 AIC3		facetted_brep ⇐ manifold_solid_brep manifold_solid_brep.outer → closed_shell facetted_brep AND brep_with_void brep_with_voids.voids[i] → closed_shell
DIRECTION	direction	AIC3	2,9,12	
FACE	face_surface	AIC3	2,9,12	
face to loop	face.bounds[i]	AIC3		face_surface ⇐ face face.bounds → loop
face to surface	face_surface.face_geometry	AIC3		face_surface face_surface.face_geometry → surface ⇒ elementary_surface ⇒ plane
FACETTED_B-REP	facetted_brep_shape_ representation	AIC3	1	
GLOBAL_UNIT	global_unit_assigned_context.units	41	8	
LOOP	poly_loop	AIC3	7,9,12	poly_loop

ARM to AIM mapping table for Facetted B-rep UoF (concluded)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
PLANE	plane	AIC3	2,9,12	
POINT	cartesian_point	AIC3	2,9,12	
POLY_LOOP	poly_loop	AIC3	7,9,12	
SHELL	closed_shell	AIC3	7	
SURFACE	plane	AIC3	2,9,12	plane
VOID	brep_with_voids.voids[i]	AIC3		

Table 5 – ARM to AIM mapping table Name preservation UoF

Application element	AIM element	Source	Rules	Reference Pa
NAME	mechanical_design_name_assignment\ name_assignment.name	204 41		mechanical_design_name_a name_assignme
name to geometric_element				mechanical_design_name .item → named_item = geometric_representat
name to topological_element				mechanical_design_name .item → named_item = topological_representa

Table 6 – ARM to AIM mapping table Product structure UoF

Application element	AIM element	Source	Rules	Reference Path
ASSEMBLY	product_definition_relationship. relating_product_definition	AIC1		{assembly_component_usage product_definition_relationshi product_definition_relationship.relatin definition
coordinate_system	shape_representation.context_ of_items	43		
user_defined_name	product.name	AIC1		product_definition_relationship.relatin definition → product_definition product_definition.version - product_version product_version.of_product product product.name
assembly to assembly				product_definition_relationship.re product_definition product_definition_relationshi product_definition_usage = assembly_component_usag

ARM to AIM mapping table Product structure UoF (continued)

Application element	AIM element	Source	Rules	Reference
assembly to coordinate_system				product_definition_rela product_de product_defi product_definition_s product_definiti shape_defin shape_definition_representa shape_definition_r shape_definition_representatio shape_represen represent representation.co
assembly to product				product_definition_rela product_de product_defi product_definit
PART	product_definition_shape	41		
user_defined_name	product_definition_shape.name	41		
part to assembly				product_defini product_definition_sh characterised_product_definitio product_definition_relation definiti
part to product				product_defini product_definition_sh characterised_product_definit product_definit

ARM to AIM mapping table Product structure UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Refere**
part to shape_representation				product_defi shape_d shape_definition_repres shape_definitio
PRODUCT	product_version	AIC1	10,11	
coordinate_system	shape_representation.context_ of_items	43		
user_defined_name	product.name	AIC1		product_versi pr produ
version_and_id	product_version.id AND product.id	AIC1 AIC1		
SHAPE_REPRESENTATION	shape_representation	41	1	
global_units	global_unit_assigned_context.units	41	3	shape_repr repres representation.c representat global_unit_a global_unit_assi
shape_representation to brep	elementary_brep_shape_rep. OR advanced_brep_shape_rep. OR facetted_brep_shape_rep.	AIC4 AIC5 AIC3	1 1 1	elementary_br shape_rep advanced_bre shape_rep facetted_brep shape_rep
TRANSFORMATION	transformation OR mapped_item	43 42 43	 2,9,12	{transformation = function cartesian_transfo

ARM to AIM mapping table Product structure UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
transformation to assembly				(transformation ← representation_relationship_with_transformation. transformation_operator representation_relationship representation_relationship.rep_2 →) (mapped_item ⇐ representation_item ← representation.items[i]) representation ⇒ shape_representation ← shape_definition_representation.representation_of shape_definition = product_definition_shape product_definition_shape.definition → characterised_product_definition = product_definition_relationship product_definition_relationship.related_ product_definition
transformation to part				(transformation ← representation_relationship_with_transformation. transformation_operator representation_relationship representation_relationship.rep_2 →) (mapped_item ⇐ representation_item ← representation.items[i]) representation ⇒ shape_representation ← shape_definition_representation.representation_of shape_definition = product_definition_shape

Table 7 – ARM to AIM mappings for visual presentation for B-rep UoF

Application element	AIM element	Source	Rules	Reference Path
3D_PROJECTION	camera_model_d3	AIC6		
3D_projection to B-rep				camera_model_d3 ⇐ camera_model_d3 ⇐ camera_model ← camera_usage.projection (DER) camera_usage ⇐ representation_map representation_map.mapped_representation representation ⇒ shape_representation ⇒ (advanced_brep_shape_representation) (elementary_brep_shape_representation) (facetted_brep_shape_representation)
3D_projection to screen_image				camera_model_d3 ⇐ camera_model ← camera_usage.projection (DER) camera_usage ← camera_image.source (DER) camera_image ⇐ geometric_representation_item ⇐ representation_item ← representation.items[i] {representation ⇒ presentation_representation ⇒} product_data_representation_view ⇒

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
				product_data_representation_vie {presentation_representati representation} ← representation_map.mapped_rep representation_map ← mapped_item.mapping_sc mapped_item ← representation.items[{representation ⇒ presentation_representatio presentation_view ⇐ {presentation_representati representation} ← representation_map.mapped_rep representation_map ← mapped_item.mapping_sc mapped_item ← representation.items[{representation ⇒ presentation_representatio presentation_area
ANNOTATION_TEXT	annotation_text_occurrence	AIC6	4	
annotation_text to screen_image				annotation_text_occurren annotation_occurrence styled_item

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
				styled_item.item {→ representation_item ⇒ geometric_representation_item ⇒ annotation_text annotation_text.text_source (DER) → annotation_text_map annotation_text_map.text_string (DER) → text_string_representation text_string_representation.strings[i] → text_or_character = (annotation_text) (text_literal_mapped_item)} ← representation.items[i] {representation ⇒ presentation_representation ⇒} view_dependent_annotation_representation ⇐ {presentation_representation ⇐ representation} ← representation_map.mapped_representation representation_map ← mapped_item.mapping_source mapped_item ← representation.items[i] {representation ⇒ presentation_representation ⇒}

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
				presentation_view ⇐ {presentation_representation ⇐ representation} ← representation_map.mapped_representatio representation_map ← mapped_item.mapping_source mapped_item ← representation.items[i] {representation ⇒ presentation_representation ⇒} presentation_area
annotation_text to transformation				annotation_text_occurrence ⇐ annotation_occurrence ⇐ styled_item styled_item.item → representation_item ⇒ geometric_representation_item ⇒ annotation_text ⇐ mapped_item {[mapped_item.mapping_target] [mapped_item.mapping_source → representation_map representation_map.mapping_origin] → {representation_item ⇒ geometric_representation_item ⇒ placement ⇒} axis2_placement_2d}
CURVE_APPEARANCE	curve_style	AIC6	5,6	

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
colour	colour_rgb	AIC6		curve_style curve_style.curve_colour → colour ⇒ colour_specification ⇒ colour_rgb
curve_font	curve_style_font	AIC6		curve_style curve_style.curve_font → curve_font_or_scaled_ curve_font_select = curve_style_font_select = curve_style_font
curve_width	curve_style.curve_width	AIC6		curve_style curve_style.curve_width
curve_appearance to curve				curve_style = presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item → representation_item ⇒ geometric_representation_item ⇒ curve
curve_appearance to edge				curve_style = presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ←

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
				styled_item.styles[i] styled_item {⇒ representation_dependent_styled_ite styled_item.item → representation_item ⇒ topological_representation_item ⇒ edge
LAYER	presentation_layer_assignment	AIC6		
user_defined_name	name_assignment.assigned_name	41		presentation_layer_assignment = named_item_select[i] ← layer_name_assignment ⇐ name_assignment name_assignment.assigned_name
layer to assembly				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → layered_item = representation ⇒ shape_representation ← shape_definition_representation. representation_model shape_definition_representation shape_definition_representation. representation_of → shape_definition = product_definition_shape

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
				product_definition_shape. definition → characterized_product_definition = product_definition_relationship product_definition_relationship. relating_product_definition
layer to B-rep				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → layered_item = representation ⇒ shape_representation ⇒ (advanced_brep_shape_representation) (elementary_brep_shape_representation) (facetted_brep_shape_representation)
layer to geometric_element				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → layered_item = geometric_representation_item
layer to layer				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → layered_item ← presentation_layer_assignment. layered_representation[i] presentation_layer_assignment

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
layer to part				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] $\rightarrow$ layered_item = representation $\Rightarrow$ shape_representation $\leftarrow$ shape_definition_representation. representation_model shape_definition_representation shape_definition_representation. representation_of $\rightarrow$ shape_definition = product_definition_shape
layer to presentation_appearance				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] $\rightarrow$ layered_item = geometric_representation_item $\Leftarrow$ representation_item $\leftarrow$ styled_item.item styled_item.styles[i] $\rightarrow$ presentation_style_assignment $\Rightarrow$ {presentation_style_by_context presentation_style_by_context.style_context style_context_select =

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
				presentation_layer_usage presentation_layer_usage.assignment → presentation_layer_assignment} presentation_style_assignment.styles[i]
layer to product				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → layered_item = representation ⇒ shape_representation ← shape_definition_representation. representation_model shape_definition_representation shape_definition_representation. representation_of → shape_definition = product_definition_shape product_definition_shape. definition → characterized_product_definition = [product_definition] [product_definition_relationship product_definition_relationship. relating_product_definition → product_definition] product_definition.version → product_version

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Path**
layer to topological_element				presentation_layer_assignment presentation_layer_assignment. layered_representation[i] → layered_item = [geometric_representation_item] [topological_representation_item]
POINT_APPEARANCE	point_style	AIC6	5,6	
colour	colour_rgb	AIC6		point_style point_style.marker_colour → colour ⇒ colour_specification ⇒ colour_rgb
marker	marker_type	AIC6		point_style point_style.marker → marker_select = marker_type
marker_size	point_style.marker_size	AIC6		point_style point_style.marker_size
point_appearance to point				point_style = presentation_style_select ← presentation_style_assignment.styles[i presentation_style_assignment ← styled_item.styles[i] styled_item ⇒ representation_dependent_styled_item representation_dependent_styled_item.iter representation_item ⇒ geometric_representation_item ⇒ point

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	**AIM element**	**Source**	**Rules**	**Reference Pa**
PRESENTATION_APPEARANCE	presentation_style_assignment. styles[i]	AIC6	5	
presentation_appearance to B-rep				presentation_style_as styles[i] presentation_style_assi styled_item.styl styled_item {= representation_dependent styled_item.item representation_ite representation.ite representation shape_representat (advanced_brep_shape_re (elementary_brep_shape_r (facetted_brep_shape_rep
presentation_appearance to shell				presentation_style_as styles[i] presentation_style_assi styled_item.styl styled_item {= representation_dependent styled_item.item representation_ite topological_representat connected_face_
SCREEN_IMAGE	presentation_area	AIC6		

ARM to AIM mappings for visual presentation for B-rep UoF (continued)

Application element	AIM element	Source	Rules	Reference Path
visual_presentation_for_B-rep UoF continued				
SURFACE_APPEARANCE	surface_style_usage	AIC6	5,6	
colour	colour_rgb	AIC6		surface_style_usage surface_style_usage.surface_side_style_s surface_style_rendering surface_style_rendering.surface_colou colour ⇒ colour_specification ⇒ colour_rgb
grid_indicator	surface_style_control_grid	AIC6		surface_style_usage surface_style_usage.surface_side_style_s (surface_style_boundary) (surface_style_control_grid) (surface_style_parameter_line) (surface_style_segmentation_curve (surface_style_silhouette_curve)
shading_method	surface_style_rendering. rendering_method	AIC6		surface_style_usage surface_style_usage.surface_side_style_s surface_style_rendering surface_style_rendering. rendering_method
surface_appearance to face				surface_style_usage = presentation_style_select ← presentation_style_assignment.style presentation_style_assignment ← styled_item.styles[i] styled_item {⇒ representation_dependent_styled_ite

ARM to AIM mappings for visual presentation for B-rep UoF (concluded)

Application element	AIM element	Source	Rules	Reference Path
				styled_item.item → representation_item ⇒ topological_representation_item ⇒ face
surface_appearance to surface				surface_style_usage presentation_style_select ← presentation_style_assignment.styles[i] presentation_style_assignment ← styled_item.styles[i] styled_item {⇒ representation_dependent_styled_item} styled_item.item → representation_item ⇒ geometric_representation_item ⇒ surface

Rules applied to table

The following rules are referenced in the preceding tables:

1. **advanced_or_elementary_or_facetted**.
2. **dependent_instantiation_of_geometry**.
3. **dependent_intantiation_of_mapped_item**.
4. **dependent_instantiation_of_pre_defined_item**.
5. **dependent_instantiation_of_presentation_style**.
6. **dependent_instantiation_of_styled_item**.
7. **dependent_instantiation_of_topology**.
8. **global_units_required**.
9. **no_complex_subtypes**.
10. **no_shape_aspects**.
11. **product_context_mechanical**.
12. **product_definition_context_design**.
13. **three_dimensional_geometry**.

5.2 AIM EXPRESS short listing

This clause specifies the *EXPRESS* schema that uses elements form the integrated resources and the AICs and contains the types, entity specializations, rules, and functions that are specific to this part of ISO 10303. This clause also specifies modifications to the textual material for the constructs that are imported from the AICs and the integrated resources. The definitions and *EXPRESS* provided in the integrated resources for constructs used in the AIM may include select list items and subtypes which are not imported into the AIM. Requirements stated in the integrated resources which refer to such items and subtypes apply exclusively to those items which are imported into the AIM.

EXPRESS specification:

```
*)
 SCHEMA part_204_brep_product_schema;
 USE FROM aic_mech_dsgn_ctxt;
 USE FROM aic_mech_dsgn_pres;
 USE FROM aic_fac_brep;
 USE FROM aic_el_brep;
 USE FROM aic_adv_brep;
 USE FROM representation_schema(representation_relationship_with_transformation,
                                functionally_defined_transformation);
 USE FROM geometry_schema(cartesian_transformation_operator_3d);
 USE FROM management_resources_schema(name_assignment);
 USE FROM support_resource_schema(label);
 USE FROM product_property_representation_schema(shape_representation);
 USE FROM product_property_definition_schema(product_definition_shape);
 USE FROM measure_schema(named_unit, si_unit,
                    conversion_based_unit,
                    global_unit_assigned_context);
(*
```

NOTES

1 – The schemas referenced above can be found in the following Parts of ISO 10303:

representation_schema	ISO 10303-43
geometry_schema	ISO 10303-42
management_resources_schema	ISO 10303-41
support_resource_schema	ISO 10303-41
measure_schema	ISO 10303-41
product_property_representation_schema	ISO 10303-41
product_property_definition_schema	ISO 10303-41
measure_schema	ISO 10303-41

2 – One further AIC is used indirectly in this AP, the **aic_adv_brep** uses the **aic_top_bnd_surf**.

3 – Part numbers are not yet assigned to the AIC documents

Definitions of types and entities unique to this AP are given below.

5.2.1 Mechanical design using boundary representation types

5.2.1.1 Mechanical design using boundary representation type definitions

5.2.1.1.1 named_item

A **named_item** is a select type that identifies the types of entities which may be given user-defined names in this AP.

EXPRESS specification:

```
*)
TYPE named_item = SELECT
     (geometric_representation_item, topological_representation_item,
     shape_representation, presentation_layer_usage);
END_TYPE;
(*
```

5.2.2 Mechanical design using boundary representation entities

5.2.2.1 Mechanical design using boundary representation entity definitions

5.2.2.1.1 mechanical_design_name_assignment

A **mechanical_design_name_assignment** allows a user-defined name to be associated with a geometric or topological element, or with a shape model, or with a layer.

EXPRESS specification:

```
*)
ENTITY mechanical_design_name_assignment
  SUBTYPE OF(name_assignment);
  item: named_item;
END_ENTITY;
(*
```

Attribute definitions:

item: the item to be named. **item** may be a geometric or topological entity, a **shape_representation**, or a **presentation_layer_usage**.

SELF\name_assignment.name: the user-defined name fo **item**.

5.2.2.2 Mechanical design using boundary representation imported entity modifications

5.2.2.2.1 solid_model

The base definition of the **solid_model** entity is given in ISO 10303-42:6.4 the following modifications apply to this part of ISO 10303.

The definition of **solid_model** is modified as follows:

A **solid_model** is a complete representation of the nominal shape of a mechanical product such that all points in the interior are connected. Any point can be calssified as being inside, outside or on the boundary of the solid.

In the context of this part of ISO 10303 a **solid_model** is a **manifold_solid_brep**.

5.2.2.2.2 geometric_representation_item

The base definition of the **geometric_representation_item** entity is given in ISO 10303-43:4.4.11 the following modifications apply to this part of ISO 10303. **Associated global rules:**
The following global rules defined in this Part of ISO 10303 apply to the **geometric_representation_item** entity:

- **dependent_instantiation_of_geometry**, see 5.2.3.2.

- **no_complex_subtypes**, see 5.2.3.11.

- **three_dimensional_geometry**, see 5.2.3.14.

5.2.2.2.3 toplogical_representation_item

The base definition of the **topological_representation_item** entity is given in ISO 10303-42:5.4.1 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **topological_representation_item** entity:

dependent_instantiation_of_topology, see 5.2.3.9.

5.2.2.2.4 mapped_item

The base definition of the **mapped_item** entity is given in ISO 10303-43:4.4.9 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **mapped_item** entity:

dependent_instantiation_of_mapped_item, see 5.2.3.3.

5.2.2.2.5 named_unit

The base definition of the **named_unit** entity is given in ISO 10303-41:23.4.1 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **named_unit** entity:

dependent_instantiation_of_named_unit, see 5.2.3.5.

5.2.2.2.6 pre_defined_item

The base definition of the **pre_defined_item** entity is given in ISO 10303-41:21.3.3 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **pre_defined_item** entity:

dependent_instantiation_of_pre_defined_item, see 5.2.3.6.

5.2.2.2.7 presentation_style_assignment

The base definition of the **presentation_style_assignment** entity is given in ISO 10303-46:6.4.4 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **presentation_style_assignment** entity:

dependent_instantiation_of_presentation_style, see 5.2.3.7.

5.2.2.2.8 product_context

The base definition of the **product_context** entity is given in ISO 10303-41:5.3.3 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **product_context** entity:

product_context_mechanical, see 5.2.3.12.

5.2.2.2.9 product_definition_context

The base definition of the **product_definition_context** entity is given in ISO 10303-41:5.3.4 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **product_definition_context** entity:

product_definition_context_design, see 5.2.3.13.

5.2.2.2.10 styled_item

The base definition of the **styled_item** entity is given in ISO 10303-46:6.4.1 the following modifications apply to this Part of ISO 10303. **Associated global rules:**

The following global rule defined in this Part of ISO 10303 applies to the **styled_item** entity:

dependent_instantiation_of_styled_item, see 5.2.3.8.

5.2.2.2.11 shape_representation

The base definition of the **shape_representation** entity is given in ISO 10303-41:8.3.1 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **shape_representation** entity:

advanced_or_elementary_or_facetted, see 5.2.3.1.

5.2.2.2.12 global_unit_assigned_context

The base definition of the **global_unit_assigned_context** entity is given in ISO 10303-41:23.4.20 the following modifications apply to this Part of ISO 10303. **Associated global rules:**
The following global rule defined in this Part of ISO 10303 applies to the **global_unit_assigned_context** entity:

global_units_required, see 5.2.3.10.

5.2.3 Mechanical design using boundary representation rule definitions

5.2.3.1 advanced_or_elementary_or_facetted

The **advanced_or_elementary_or_facetted** rule ensures that any instance of a **shape_representation** conforms to the specification of one of:

- an **advanced_brep_shape_representation**;
- an **elementary_brep_shape_representation**;
- a **facetted_brep_shape_representation**.

EXPRESS specification:

```
*)
RULE
advanced_or_elementary_or_facetted FOR(shape_representation);
WHERE
   WR1: SIZEOF (QUERY (sr <* shape_representation |
                NOT( SIZEOF(
       ['PART_204_BREP_PRODUCT_SCHEMA.ADVANCED_BREP_SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_BREP_SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.FACETTED_BREP_SHAPE_REPRESENTATION']
           * TYPEOF (sr)) =1 ))) = 0;
END_RULE;
```

(*

Argument definitions:

shape_representation: identifies the set of all instances of **shape_representation** entities to which the rule is applied.

Formal propositions:

WR1: Each instance of **shape_representation** shall be either an **advanced_brep_shape_-representation** or an **elementary_brep_shape_representation** or a **facetted_brep_shape_-representation**.

5.2.3.2 dependent_instantiation_of_geometry

This rule ensures that any instance of a **geometric_representation_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_geometry FOR (geometric_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (gri <* geometric_representation_item |
                 NOT (SIZEOF (USEDIN (gri,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

geometric_representation_item: identifies the set of all instances of **geometric_represent-ation_item** subtypes.

Formal propositions:

WR1: Any instance of **geometric_representation_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.3 dependent_instantiation_of_mapped_item

This rule ensures that any instance of a **mapped_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_mapped_item FOR (mapped_item);
   WHERE
      WR1 : SIZEOF ( QUERY (mi <* mapped_item |
                 NOT (SIZEOF (USEDIN (mi,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

mapped_item: identifies the set of all instances of **mapped_item** subtypes.

Formal propositions:

WR1: Any instance of **mapped_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.4 dependent_instantiation_of_mechanical_design_presentation_represe

This rule ensures that any instance of a **mechanical_design_presentation_representation** is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_mechanical_design_presentation_representation
   FOR (mechanical_design_presentation_representation);
WHERE
  WR1 : SIZEOF ( QUERY (mdpr <* mechanical_design_presentation_representation |
        NOT (SIZEOF (USEDIN (mdpr,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

mechanical_design_presentation_representation: identifies the set of all instances of **mechanical_design_presentation_representation**.

Formal propositions:

WR1: Any instance of a **mechanical_design_presentation_representation** shall be used in the definition of some other entity.

5.2.3.5 dependent_instantiation_of_named_unit

This rule ensures that any instance of a **named_unit** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_named_unit FOR (named_unit);
   WHERE
      WR1 : SIZEOF ( QUERY (nmu <* named |
                NOT (SIZEOF (USEDIN (nmu,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

named_unit: identifies the set of all instances of **named_unit** subtypes.

Formal propositions:

WR1: Any instance of **named_unit** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.6 dependent_instantiation_of_pre_defined_item

This rule ensures that any instance of a **pre_defined_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_pre_defined_item FOR
                   (pre_defined_item);
   WHERE
      WR1 : SIZEOF ( QUERY (pdi <* pre_defined_item |
                 NOT (SIZEOF (USEDIN (pdi,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

pre_defined_item: identifies the set of all instances of **pre_defined_item** subtypes.

Formal propositions:

WR1: Any instance of **pre_defined_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.7 dependent_instantiation_of_presentation_style

This rule ensures that any instance of a **presentation_style_assignment** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_presentation_style FOR
                   (presentation_style_assignment);
   WHERE
      WR1 : SIZEOF ( QUERY (psa <* presentation_style_assignment |
                 NOT (SIZEOF (USEDIN (psa,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

presentation_style_assignment: identifies the set of all instances of **presentation_style_-assignment** subtypes.

Formal propositions:

WR1: Any instance of **presentation_style_assignment** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.8 dependent_instantiation_of_styled_item

This rule ensures that any instance of a **styled_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_styled_item FOR (styled_item);
   WHERE
      WR1 : SIZEOF ( QUERY (si <* styled_item |
                 NOT (SIZEOF (USEDIN (si,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

styled_item: identifies the set of all instances of **styled_item** subtypes.

Formal propositions:

WR1: Any instance of **styled_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.9 dependent_instantiation_of_topology

This rule ensures that any instance of a **topological_representation_item** which is defined is used as part of the definition of some other entity.

EXPRESS specification:

```
*)
RULE dependent_instantiation_of_topology FOR (topological_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (tri <* topological_representation_item |
                 NOT (SIZEOF (USEDIN (tri,'')) > 0 ))) = 0;
END_RULE;
(*
```

Argument definitions:

topological_representation_item: identifies the set of all instances of **topological_representation_item** subtypes.

Formal propositions:

WR1: Any instance of **topological_representation_item** which occurs in this application protocol shall be used in the definition of some other entity.

5.2.3.10 global_units_required

The **global_units_required** rule specifies the units that shall be defined for **global_unit_assigned_context**. Every **global_unit_assigned_context** shall have a maximum of 2 elements in its set of units, these shall include a unit of length and may also include a unit of plane angle measure.

EXPRESS specification:

```
*)
RULE global_units_required FOR (global_unit_assigned_context);
   WHERE
      WR1 : SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                 NOT (SIZEOF (guac.units) <= 2))) = 0;
```

```
      WR2 : SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                 NOT (SIZEOF (QUERY( u <* guac.units |
                   'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF
                   (u))) =1 ))) = 0;
      WR3:  SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                 NOT (SIZEOF (QUERY( u <* guac.units |
                 NOT (SIZEOF (QUERY (other_u <* guac.units - u |
                  'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF(u))
                  AND
                   'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_UNIT]' IN
                       TYPEOF (other_u))))) = 0))) = 0;
END_RULE;
(*
```

Argument definitions:

global_unit_assigned_context: identifies the set of all instances of **global_unit_assigned_-context** entities.

Formal propositions:

WR1: For any instance of **global_unit_assigned_context** which occurs in this application protocol the set of units shall contain 2 or less elements.

WR2: For any instance of **global_unit_assigned_context** which occurs in this application protocol the set of units shall contain a **length_unit**.

WR3: For any instance of **global_unit_assigned_context** which occurs in this application protocol the set of units shall contain only **length_unit**s or **plane_angle_unit**s.

Informal propositions:

IP1: Any entity instance containing a **length_measure** as part of its definition shall be only occur in a **global_unit_assigned_context**.

IP2: Any entity instance containing a **plane_angle_measure** as part of its definition shall be only occur in a **global_unit_assigned_context** for which a **plane_angle_unit** is defined.

5.2.3.11 no_complex_subtypes

The following rule is equivalent to a ONEOF declaration and ensures that no subtype of **geometric_representation_item** can simultaneously be of more than one of the subtypes listed in the rule.

EXPRESS specification:

```
 *)
 RULE no_complex_subtypes FOR(geometric_representation_item);
   WHERE
    WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
    NOT (SIZEOF (TYPEOF(gri) * ['PART_204_BREP_PRODUCT_SCHEMA.ANNOTATION_TEXT',
               'PART_204_BREP_PRODUCT_SCHEMA.CURVE',
                'PART_204_BREP_PRODUCT_SCHEMA.DIRECTION',
          'PART_204_BREP_PRODUCT_SCHEMA.EDGE_CURVE',
```

```
           'PART_204_BREP_PRODUCT_SCHEMA.FACE_SURFACE',
            'PART_204_BREP_PRODUCT_SCHEMA.PLACEMENT',
             'PART_204_BREP_PRODUCT_SCHEMA.POINT',
           'PART_204_BREP_PRODUCT_SCHEMA.POLY_LOOP',
             'PART_204_BREP_PRODUCT_SCHEMA.SOLID_MODEL',
              'PART_204_BREP_PRODUCT_SCHEMA.SURFACE',
               'PART_204_BREP_PRODUCT_SCHEMA.VECTOR',
           'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT']) < 2 ))) = 0;
 END_RULE;
 (*
```

Argument definitions:

geometric_representation_item: identifies the set of all instances of **geometric_representation_item** entities to which the rule is applied.

Formal propositions:

WR1: The TYPEOF function shall not return two or more of the subtypes listed in this rule for any **geometric_representation_item** used in this AP.

5.2.3.12 product_context_mechanical

In the context of this AP the only valid subtype of **product_context** is **mechanical_context**.

EXPRESS specification:

```
*)
RULE product_context_mechanical FOR (product_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pc <* product_context |
                 NOT ('PART_204_BREP_PRODUCT_SCHEMA.MECHANICAL_CONTEXT'
            IN TYPEOF(pc)))) = 0;
END_RULE;
(*
```

Argument definitions:

product_context: identifies the set of all instances of **product_context** entities.

Formal propositions:

WR1: Any instance of **product_context** which occurs in this application protocol shall be of the **mechanical_context** subtype.

5.2.3.13 product_definition_context_design

In the context of this AP the only valid **product_definition_context**s are of the special subtype **design_context**.

EXPRESS specification:

```
*)
RULE product_definition_context_design FOR (product_definition_context);
   WHERE
```

```
      WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
                 NOT ('PART_204_BREP_PRODUCT_SCHEMA.DESIGN_CONTEXT'
                   IN TYPEOF(pc)))) = 0;
END_RULE;
(*
```

Argument definitions:

product_definition_context: identifies the set of all instances of **product_definition_context** entities.

Formal propositions:

WR1: Any instance of **product_definition_context** which occurs in this application protocol shall be of subtype **design_context**.

5.2.3.14 three_dimensional_geometry

The following rule ensures that all geometry used in the definition of a B-rep product is three dimensional or is one of the specialised entities used in the presentation of a two dimensional image. All geometric representation items have a derived integer attribute dim which gives the dimension of the space in which they are defined. This rule verifies that 2 dimensional geometry is not used in the definition of geometric models.

EXPRESS specification:

```
*)
RULE three_dimensional_geometry FOR(geometric_representation_item);
  WHERE
   WR1: SIZEOF(QUERY(gri <* geometric_representation_item |
           (gri.dim = 2) AND NOT (SIZEOF (
           ['PART_204_BREP_PRODUCT_SCHEMA.PLANAR_BOX',
           'PART_204_BREP_PRODUCT_SCHEMA.ANNOTATION_TEXT'] *
             TYPEOF(gri)) =1 ))) = 0;
END_RULE;
(*
```

Argument definitions:

geometric_representation_item: identifies the set of all instances of **geometric_representation_item** entities to which the rule is applied.

Formal propositions:

WR1: Each instance of **geometric_representation_item** occurring in this AP shall have a dimension of 3 or be of type **planar_box** or of type **annotation_text**.

EXPRESS specification:

```
*)
END_SCHEMA; --end part 204 brep product schema;
(*
```

6 Conformance Requirements

Conformance to this part of ISO 10303 includes satisfying the requirements stated in this part, the requirements of the implementation method(s) supported, and the relevant requirements of the normative references.

An implementation shall support at least one of the following implementation methods: ISO 10302-21, ISO 10303-22. Requirements with respect to implementation methods are specified in annex D.

The Protocol Information Conformance Statement (PICS) proforma lists the options or the combination of options that may be included in the implementation. The PICS proforma is provided in annex C.

This Part of ISO 10303 provides for a number of options that may be supported by an implementation. These options have been grouped into the following conformance classes.

Class 1: B-rep level 1: The definition of a mechanical engineering product model where the shape is represented by one or more facetted_brep_models.

Class 2: B-rep level 2: The definition of a mechanical engineering product model where the shape is represented by one or more elementary_brep_models.

Class 3: B-rep level 3: The definition of a mechanical engineering product model where the shape is represented by one or more advanced_brep_models.

Support for a particular conformance class requires support for all the options specified in that class.

NOTE 1 – ISO 10303-1204 defines the abstract test suite and test purposes to be used in the assessment of conformance.

General requirements for all classes are:

a) The information requirements and relationships of the ARM shall be preserved in the implementation. This includes support for all valid combinations of entities and their attributes. No 'substitution' of entities shall be permitted. Consequently all construct assertions from clause 4 shall be maintained.

b) All AIM entities, types, and their associated constraints shall be supported. Treatment of options and default values shall conform to the AIM.

c) All AIM entities, types, and their associated constraints shall be read and processed by a postprocessor.

d) Entities not belonging to the AIM shall be excluded from a preprocessor implementation. Entities not specified in the AIM shall not be included in a conforming exchange structure.

e) An implementation shall satisfy all general requirements of implementation methods (given in the appropriate 20-series class). This includes the preservation of the AIM mapped onto the implementation form and conformance to the syntax of the implementation form.

Table 8 – units of functionality within conformance classes

	Class 1	Class 2	Class 3
name preservation	O	O	O
product structure	R	R	R
visual presentation for B-rep	O	O	O
facetted B-rep	R		
elementary B-rep		R	
advanced B-rep			R

NOTE 2 – In table 8 O denotes an optional capability which may be supported by the pre-processor and shall be supported by all post-processors. R denotes a requirement on both pre- and post-processor.

Annex A
(normative)

AIM EXPRESS long listing

This Annex includes all longforms related to the different AICs used in this document.

A.1 AIM EXPRESS listing

(* Manually extracted EXPRESS listing for Part 204 Express from presentation AIC Any changes in supertypes, select types etc resulting from above.

```
*)
 SCHEMA part_204_brep_product_schema

 CONSTANT
   aic_advanced_brep_functionality_definition : LIST [2:2] OF STRING
    := ['aic_adv_brep', 'the definition of a B-rep model with explicitly ' +
      'defined geometry, elementary or sculptured surfaces ' +
      'and topological boundaries'];

   aic_topology_bounded_surface_functionality_definition : LIST [2:2] OF STRING
    := ['aic_top_bnd_surf', 'the definition of a face with explicitly' +
      'defined geometry and topological boundaries'];

  aic_elementary_brep_functionality_definition : LIST [2:2] OF STRING
   := ['aic_el_brep', 'the definition of a B-rep model with explicitly ' +
     'defined geometry, simple curves, elementary surfaces ' +
     'and topological boundaries'];

   aic_facetted_brep_functionality_definition : LIST [2:2] OF STRING :=
     ['aic_fac_brep', 'the definition of a facetted B-rep shape ' +
      'representation consisting of one or more facetted B-reps with ' +
      'planar faces bounded by poly loops'];

   mechanical_design_context_aic_function_definition : LIST [2:2] OF STRING
     := ['aic_mech_design', 'product identification and relation for the' +
      'definition of mechanical product within the design phase of the' +
      'product life cycle'];

   aic_mech_dsgn_pres_fd : LIST [2:2] OF STRING :=
      ['aic_mech_dsgn_pres', 'the projection from some mechanical design'+
       ' using either surface or boundary representation to'+
       ' an associated annotated picture on a screen'];
 END_CONSTANT;

TYPE axis2_placement = SELECT
   (axis2_placement_2d,
    axis2_placement_3d);
 END_TYPE;


 TYPE bspline_curve_form = ENUMERATION OF (
   polyline_form,
   circular_arc,
   elliptic_arc,
   parabolic_arc,
   hyperbolic_arc,
   unspecified);
 END_TYPE;

 TYPE bspline_surface_form = ENUMERATION OF
```

```
   (plane_surf,
    cylindrical_surf,
    conical_surf,
    spherical_surf,
    toroidal_surf,
    surf_of_revolution,
    ruled_surf,
    generalised_cone,
    quadric_surf,
    surf_of_linear_extrusion,
    unspecified);
 END_TYPE;

 TYPE dimension_count = INTEGER;
 WHERE
   WR1: SELF > 0;
 END_TYPE;

TYPE knot_type = ENUMERATION OF
   (uniform_knots,
    unspecified,
    quasi_uniform_knots,
    piecewise_bezier_knots);
 END_TYPE;

TYPE label = STRING;
END_TYPE;

TYPE length_measure = REAL;
END_TYPE;

TYPE measure_value = SELECT
  (length_measure,
   plane_angle_measure,
   parameter_value,
   positive_length_measure);
END_TYPE;


 TYPE named_item = SELECT
      (geometric_representation_item, topological_representation_item,
      shape_representation, presentation_layer_usage);
 END_TYPE

TYPE plane_angle_measure = REAL;
END_TYPE;

TYPE positive_length_measure = length_measure;
WHERE
  WR1: SELF > 0;
END_TYPE;
```

```
 TYPE parameter_value = REAL;
 END_TYPE;

TYPE reversible_topology_item = SELECT
   (face,
    face_bound,
    closed_shell);
 END_TYPE;

 TYPE reversible_topology = SELECT
   (reversible_topology_item,
   set_of_reversible_topology_item);
 END_TYPE;

  TYPE set_of_reversible_topology_item
         = SET [0:?] of reversible_topology_item;
  END_TYPE;

TYPE si_unit_name = ENUMERATION OF
  (metre,
   radian);
END_TYPE;

TYPE si_prefix = ENUMERATION OF
  (exa,
   peta,
   tera,
   giga,
   mega,
   kilo,
   hecto,
   deca,
   deci,
   centi,
   milli,
   micro,
   nano,
   pico,
   femto,
   atto);
END_TYPE;

TYPE text = STRING;
END_TYPE;

TYPE transformation = SELECT
  (functionally_defined_transformation);
END_TYPE;

TYPE unit = SELECT
  (named_unit,
```

```
   derived_unit);
END_TYPE;

 TYPE vector_or_direction = SELECT
   (vector,
    direction);
 END_TYPE;

TYPE year_number = INTEGER;
END_TYPE;


(* Types from presentation AIC - to be verified *)

TYPE area_or_view = SELECT
       (presentation_area,
        presentation_view);
END_TYPE;

TYPE central_or_parallel = ENUMERATION OF (central, parallel);
END_TYPE;

TYPE curve_font_or_scaled_curve_font_select = SELECT(
                        curve_style_font_select);
END_TYPE;

TYPE curve_or_render = SELECT (
   curve_style,
   curve_style_rendering);
END_TYPE;

TYPE curve_style_font_select = SELECT(
         curve_style_font);
END_TYPE;

TYPE direction_count_select = SELECT (
      u_direction_count, v_direction_count);
END_TYPE;

TYPE font_select = SELECT(predefined_text_font);
END_TYPE;

TYPE layered_item = SELECT (
        representation,
        geometric_representation_item);
WHERE
   WR1: NOT ('AIC_MECH_DSGN_PRES.'+
             'REPRESENTATION_ITEM_WITHOUT_STYLE'
             IN TYPEOF(SELF));
END_TYPE;
```

```
TYPE marker_select = SELECT(marker_type);
END_TYPE;

TYPE marker_type = ENUMERATION OF (dot, x, plus, asterisk, circle,
                                   square, triangle);
END_TYPE;

TYPE null_style = ENUMERATION OF (null);
END_TYPE;

TYPE presentable_text = STRING;
END_TYPE;

TYPE presentation_size_assigment_select = SELECT (
          presentation_area,
          presentation_view);
END_TYPE;

TYPE presentation_style_select = SELECT(
     predefined_presentation_style,
     point_style,
     curve_style,
     surface_style_usage,
     null_style);
END_TYPE;

TYPE shading_curve_method = ENUMERATION OF (constant_colour, linear_colour);
END_TYPE;

TYPE shading_surface_method = ENUMERATION OF (constant_shading,
                                              colour_shading,
                                              dot_shading,
                                              normal_shading);
END_TYPE;

TYPE size_select  = SELECT(positive_length_measure,
                           measure_with_unit);
END_TYPE;

TYPE style_context_select = SELECT (
         presentation_area ,
         presentation_view ,
         product_data_representation_view ,
         presentation_layer_usage);
END_TYPE;

TYPE surface_side = ENUMERATION OF(positive, negative, both);
END_TYPE;

TYPE surface_side_style_select = SELECT(
          surface_side_style);
END_TYPE;
```

```
TYPE surface_style_element_select = SELECT (
   surface_style_rendering,
   surface_style_boundary,
   surface_style_silhouette,
   surface_style_segmentation_curve,
   surface_style_control_grid,
   surface_style_parameter_line);
END_TYPE;

TYPE text_alignment = label;
END_TYPE;

TYPE text_or_character = SELECT(
   annotation_text,
   text_literal_mapped_item);
END_TYPE;

TYPE text_path = ENUMERATION OF (left, right, up, down);
END_TYPE;

TYPE u_direction_count = INTEGER;
WHERE
   WR1: SELF > 1;
END_TYPE;

TYPE v_direction_count = INTEGER;


 TYPE shell = SELECT (closed_shell);
 END_TYPE;

TYPE list_of_reversible_topology_item =
                               LIST [0:?] of reversible_topology_item;
 END_TYPE;


ENTITY name_assignment
  ABSTRACT SUPERTYPE;
  assigned_name : label;
END_ENTITY;

ENTITY mechanical_design_name_assignment
   SUBTYPE OF(name_assignment);
   item: named_item;
END_ENTITY

ENTITY geometric_representation_item
  SUBTYPE OF (representation_item);
DERIVE
  dim : dimension_count := dimension_of(SELF);
END_ENTITY;
```

```
ENTITY point
  SUPERTYPE OF (ONEOF(cartesian_point))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY cartesian_point
  SUBTYPE OF (point);
  coordinates  : LIST [1:3] OF length_measure;
END_ENTITY;

ENTITY direction
  SUBTYPE OF (geometric_representation_item);
  direction_ratios : LIST [2:3] OF REAL;
WHERE
  WR1: NOT((direction_ratios[1] = 0.0) AND
           (direction_ratios[2] = 0.0) AND
           (direction_ratios[3] = 0.0));
END_ENTITY;

ENTITY vector
  SUBTYPE OF (geometric_representation_item);
  orientation : direction;
  magnitude   : length_measure;
WHERE
  WR1 : magnitude >= 0.0;
END_ENTITY;

ENTITY placement
  SUPERTYPE OF (ONEOF(axis1_placement, axis2_placement_3d))
  SUBTYPE OF (geometric_representation_item);
  location : cartesian_point;
END_ENTITY;

ENTITY axis1_placement
  SUBTYPE OF (placement);
  axis     : OPTIONAL direction;
DERIVE
  z        : direction := NVL(normalise(axis), direction([0.0,0.0,1.0]));
WHERE
  WR1: SELF\geometric_representation_item.dim  = 3;
END_ENTITY;

ENTITY axis2_placement_2d
  SUBTYPE OF (placement);
  ref_direction : OPTIONAL direction;
DERIVE
  p             : LIST [2:2] OF direction := build_2axes(ref_direction);
WHERE
  WR1: SELF\geometric_representation_item.dim = 2;
END_ENTITY;
```

```
 ENTITY axis2_placement_3d
   SUBTYPE OF (placement);
   axis          : OPTIONAL direction;
   ref_direction : OPTIONAL direction;
 DERIVE
   p             : LIST [3:3] OF direction := build_axes(axis,ref_direction);
 WHERE
   WR1: SELF\placement.location.dim = 3;
   WR2: (NOT EXISTS (axis) OR axis.dim = 3);
   WR3: (NOT EXISTS (ref_direction) OR ref_direction.dim = 3);
   WR4: NOT EXISTS (axis) OR NOT EXISTS (ref_direction) OR
          (cross_product(axis,ref_direction).magnitude > 0.0);
 END_ENTITY;

ENTITY functionally_defined_transformation;
END_ENTITY;

ENTITY cartesian_transformation_operator
   SUPERTYPE OF(ONEOF(cartesian_transformation_operator_3d))
   SUBTYPE OF (geometric_representation_item,
                         functionally_defined_transformation);
   axis1        : OPTIONAL direction;
   axis2        : OPTIONAL direction;
   local_origin : cartesian_point;
   scale        : OPTIONAL REAL;
 DERIVE
   scl          : REAL := NVL(scale, 1.0);
 WHERE
   WR1: scl > 0.0;
 END_ENTITY;

ENTITY cartesian_transformation_operator_3d
   SUBTYPE OF (cartesian_transformation_operator);
   axis3 : OPTIONAL direction;
 DERIVE
   u     : LIST[3:3] OF direction
         := base_axis(3,SELF\cartesian_transformation_operator.axis1,
                        SELF\cartesian_transformation_operator.axis2,axis3);
 WHERE
   WR1: SELF\cartesian_transformation_operator.dim = 3;
 END_ENTITY;


 ENTITY curve
   SUPERTYPE OF (ONEOF(line, conic, bounded_curve))
   SUBTYPE OF (geometric_representation_item);
 END_ENTITY;

 ENTITY line
   SUBTYPE OF (curve);
   pnt : cartesian_point;
```

```
  dir : vector;
WHERE
  WR1: dir.dim  = pnt.dim;
END_ENTITY;

ENTITY conic
  ABSTRACT SUPERTYPE OF (ONEOF(circle, ellipse, hyperbola, parabola))
  SUBTYPE OF (curve);
  position: axis2_placement;
END_ENTITY;

ENTITY circle
  SUBTYPE OF (conic);
  radius   : positive_length_measure;
END_ENTITY;

ENTITY ellipse
  SUBTYPE OF (conic);
  semi_axis_1 : positive_length_measure;
  semi_axis_2 : positive_length_measure;
END_ENTITY;

ENTITY hyperbola
  SUBTYPE OF (conic);
  semi_axis      : positive_length_measure;
  semi_imag_axis : positive_length_measure;
END_ENTITY;

ENTITY parabola
  SUBTYPE OF (conic);
  focal_dist : length_measure;
WHERE
  WR1: focal_dist <> 0.0;
END_ENTITY;

ENTITY bounded_curve
  SUPERTYPE OF (ONEOF(polyline, b_spline_curve))
  SUBTYPE OF (curve);
END_ENTITY;

ENTITY polyline
  SUBTYPE OF (bounded_curve);
  points : LIST [2:?] OF cartesian_point;
END_ENTITY;

ENTITY b_spline_curve
  SUPERTYPE OF (ONEOF(uniform_curve, quasi_uniform_curve, bezier_curve,
                    b_spline_curve_with_knots)ANDOR rational_b_spline_curve)
  SUBTYPE OF (bounded_curve);
  degree               : INTEGER;
  control_points_list  : LIST [2:?] OF cartesian_point;
  curve_form           : bspline_curve_form;
```

```
   closed_curve         : LOGICAL;
   self_intersect       : LOGICAL;
 DERIVE
   upper_index_on_control_points  : INTEGER
                                  := (SIZEOF(control_points_list) - 1);
   control_points       : ARRAY [0:upper_index_on_control_points]
                                                      OF cartesian_point
                                  := list_to_array(control_points_list,0,
                                             upper_index_on_control_points);
 WHERE
   WR1: ('PART_204_BREP_PRODUCT_SCHEMA.UNIFORM_CURVE' IN TYPEOF(self)) OR
      ('PART_204_BREP_PRODUCT_SCHEMA.QUASI_UNIFORM_CURVE' IN TYPEOF(self)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.BEZIER_CURVE' IN TYPEOF(self)) OR
        ('PART_204_BREP_PRODUCT_SCHEMA.B_SPLINE_CURVE_WITH_KNOTS' IN
                                                        TYPEOF(self));
 END_ENTITY;

 ENTITY b_spline_curve_with_knots
   SUBTYPE OF (b_spline_curve);
   knot_multiplicities  : LIST [2:?] OF INTEGER;
   knots                : LIST [2:?] OF parameter_value;
   knot_spec            : knot_type;
 DERIVE
   upper_index_on_knots : INTEGER := SIZEOF(knots);
 WHERE
   WR1: constraints_param_bspl(degree, upper_index_on_knots,
                               upper_index_on_control_points,
                               knot_multiplicities, knots);
    WR2: SIZEOF(knot_multiplicities) = upper_index_on_knots;
END_ENTITY;

 ENTITY uniform_curve
   SUBTYPE OF (b_spline_curve);
 END_ENTITY;

 ENTITY quasi_uniform_curve
   SUBTYPE OF (b_spline_curve);
 END_ENTITY;



 ENTITY bezier_curve
   SUBTYPE OF (b_spline_curve);
 END_ENTITY;

 ENTITY rational_b_spline_curve
   SUBTYPE OF (b_spline_curve);
   weights_data : LIST [2:?] OF REAL;

 DERIVE
   weights              : ARRAY [0:upper_index_on_control_points] OF REAL
                                  := list_to_array(weights_data,0,
                                         upper_index_on_control_points);
```

```
WHERE
  WR1:  SIZEOF(weights_data) = SIZEOF(SELF\b_spline_curve.control_points_list);
  WR2:  curve_weights_positive(SELF);
END_ENTITY;



ENTITY surface
  SUPERTYPE OF (ONEOF(elementary_surface, swept_surface, bounded_surface))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY elementary_surface
  SUPERTYPE OF (ONEOF(plane, cylindrical_surface, conical_surface,
                      spherical_surface, toroidal_surface))
  SUBTYPE OF (surface);
  position : axis2_placement_3d;
END_ENTITY;

ENTITY plane
SUBTYPE OF (elementary_surface);
END_ENTITY;

ENTITY
cylindrical_surface
  SUBTYPE OF (elementary_surface);
  radius : positive_length_measure;
END_ENTITY;

ENTITY
conical_surface
  SUBTYPE OF (elementary_surface);
  radius     : length_measure;
  semi_angle : plane_angle_measure;
WHERE
  WR1: radius >= 0.0;
END_ENTITY;

ENTITY spherical_surface
  SUBTYPE OF (elementary_surface);
  radius   : positive_length_measure;
END_ENTITY;

ENTITY toroidal_surface
  SUBTYPE OF (elementary_surface);
  major_radius : positive_length_measure;
  minor_radius : positive_length_measure;
WHERE
 WR1: major_radius > minor_radius;
END_ENTITY;

ENTITY swept_surface
  SUPERTYPE OF (ONEOF(surface_of_linear_extrusion, surface_of_revolution))
```

```
  SUBTYPE OF (surface);
  swept_curve : curve;
END_ENTITY;

ENTITY surface_of_linear_extrusion
  SUBTYPE OF (swept_surface);
  extrusion_axis      : vector;
END_ENTITY;

ENTITY surface_of_revolution
  SUBTYPE OF (swept_surface);
  axis_position       : axis1_placement;
DERIVE
  axis_line : line := line(axis_position.location,axis_position.z);
END_ENTITY;

ENTITY bounded_surface
  SUPERTYPE OF (ONEOF(b_spline_surface))
  SUBTYPE OF (surface);
END_ENTITY;

ENTITY b_spline_surface
  SUPERTYPE OF (ONEOF(uniform_surface, quasi_uniform_surface, bezier_surface,
              b_spline_surface_with_knots)ANDOR rational_b_spline_surface)
  SUBTYPE OF (bounded_surface);
  u_degree             : INTEGER;
  v_degree             : INTEGER;
  control_points_list  : LIST [2:?] OF
                           LIST [2:?] OF cartesian_point;
  surface_form         : bspline_surface_form;
  u_closed             : LOGICAL;
  v_closed             : LOGICAL;
  self_intersect       : LOGICAL;
DERIVE
  u_upper             : INTEGER := SIZEOF(control_points_list) - 1;
  v_upper             : INTEGER := SIZEOF(control_points_list[1]) - 1;
  control_points      : ARRAY [0:u_upper] OF ARRAY [0:v_upper] OF
                        cartesian_point
                     := make_array_of_array(control_points_list,
                                              0,u_upper,0,v_upper);
WHERE
  WR1: ('PART_204_BREP_PRODUCT_SCHEMA.UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
    ('PART_204_BREP_PRODUCT_SCHEMA.QUASI_UNIFORM_SURFACE' IN TYPEOF(SELF)) OR
       ('PART_204_BREP_PRODUCT_SCHEMA.BEZIER_SURFACE' IN TYPEOF(SELF)) OR
       ('PART_204_BREP_PRODUCT_SCHEMA.BSPLINE_SURFACE_WITH_KNOTS' IN
                                                        TYPEOF(SELF));
END_ENTITY;


ENTITY b_spline_surface_with_knots
  SUBTYPE OF (b_spline_surface);
  u_multiplicities  : LIST [2:?] OF INTEGER;
```

```
  v_multiplicities  : LIST [2:?] OF INTEGER;
  u_knots           : LIST [2:?] OF parameter_value;
  v_knots           : LIST [2:?] OF parameter_value;
  knot_spec         : knot_type;
DERIVE
  knot_u_upper      : INTEGER := SIZEOF(u_knots);
  knot_v_upper      : INTEGER := SIZEOF(v_knots);
WHERE
   WR1: constraints_param_bspl(SELF\b_spline_surface.u_degree,
                  knot_u_upper, SELF\b_spline_surface.u_upper,
                              u_multiplicities, u_knots);
   WR2: constraints_param_bspl(SELF\b_spline_surface.v_degree,
                  knot_v_upper, SELF\b_spline_surface.v_upper,
                              v_multiplicities, v_knots);
   WR3: SIZEOF(u_multiplicities) = knot_u_upper;
   WR4: SIZEOF(v_multiplicities) = knot_v_upper;
END_ENTITY;

ENTITY uniform_surface
  SUBTYPE OF (b_spline_surface);
END_ENTITY;

ENTITY quasi_uniform_surface
  SUBTYPE OF (b_spline_surface);
DERIVE
  ku_up : INTEGER := u_upper - u_degree + 2;
  kv_up : INTEGER := v_upper - v_degree + 2;
END_ENTITY;

ENTITY bezier_surface
  SUBTYPE OF (b_spline_surface);
DERIVE
  ku_up : INTEGER := (u_upper/u_degree) + 1;
  kv_up : INTEGER := (v_upper/v_degree) + 1;
END_ENTITY;

ENTITY rational_b_spline_surface
  SUBTYPE OF (b_spline_surface);
  weights_data : LIST [2:?] OF
                   LIST [2:?] OF REAL;

DERIVE
  weights       : ARRAY [0:u_upper] OF
                     ARRAY [0:v_upper] OF REAL
                := make_array_of_array(weights_data,0,u_upper,0,v_upper);
WHERE
  WR1: SIZEOF(weights_data) = SIZEOF(SELF\b_spline_surface.control_points_list)
          AND SIZEOF(weights_data[1]) =
                          SIZEOF(SELF\b_spline_surface.control_points_list[1]);
  WR2: surface_weights_positive(SELF);
END_ENTITY;
```

```
ENTITY topological_representation_item
  SUPERTYPE OF (ONEOF(vertex, edge, face_bound, face, connected_face_set,
                                                      (path ANDOR loop)))
  SUBTYPE OF (representation_item);
END_ENTITY;

ENTITY vertex
  SUBTYPE OF (topological_representation_item);
END_ENTITY;

ENTITY vertex_point
  SUBTYPE OF(vertex,geometric_representation_item);
    vertex_geometry : point;
END_ENTITY;

ENTITY edge
  SUPERTYPE OF(ONEOF(edge_curve, oriented_edge))
  SUBTYPE OF (topological_representation_item);
  edge_start : vertex;
  edge_end   : vertex;
END_ENTITY;

ENTITY edge_curve
  SUBTYPE OF(edge,geometric_representation_item);
  edge_geometry : curve;
  same_sense    : BOOLEAN;
END_ENTITY;


ENTITY oriented_edge
  SUBTYPE OF (edge);
  edge_element : edge;
  orientation  : BOOLEAN;
DERIVE
  edge_start   : vertex := boolean_choose (SELF.orientation,
                                           SELF.edge_element.edge_start,
                                           SELF.edge_element.edge_end);
  edge_end     : vertex := boolean_choose (SELF.orientation,
                                           SELF.edge_element.edge_end,
                                           SELF.edge_element.edge_start);
WHERE
  WR1: NOT ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_EDGE' IN
                                       TYPEOF (SELF.edge_element));
END_ENTITY;

ENTITY path
  SUPERTYPE OF (ONEOF(edge_loop, oriented_path))
  SUBTYPE OF (topological_representation_item);
  edge_list  : LIST [1:?] OF UNIQUE oriented_edge;
WHERE
  WR1: path_head_to_tail(SELF);
END_ENTITY;
```

```
 ENTITY loop
   SUPERTYPE OF (ONEOF(vertex_loop, edge_loop, poly_loop))
   SUBTYPE OF (topological_representation_item);
 END_ENTITY;



 ENTITY vertex_loop
   SUBTYPE OF (loop);
   loop_vertex : vertex;
 END_ENTITY;

 ENTITY edge_loop
   SUBTYPE OF (loop,path);
 DERIVE
   ne : INTEGER := SIZEOF(SELF\path.edge_list);
 WHERE
   WR1: (SELF\path.edge_list[1].edge_element.edge_start) :=:
        (SELF\path.edge_list[ne].edge_element.edge_end);
 END_ENTITY;

 ENTITY poly_loop
   SUBTYPE OF (loop,geometric_representation_item);
   polygon : LIST [3:?] OF UNIQUE cartesian_point;
 END_ENTITY;

ENTITY face_bound
   SUBTYPE OF(topological_representation_item);
   bound       :  loop;
   orientation :  BOOLEAN;
 END_ENTITY;

  ENTITY face_outer_bound
    SUBTYPE OF (face_bound);
  END_ENTITY;

ENTITY face
   SUPERTYPE OF(ONEOF(face_surface, oriented_face))
   SUBTYPE OF (topological_representation_item);
   bounds : SET[1:?] OF face_bound;
 WHERE
   WR1: NOT (mixed_loop_type_set(list_to_set(list_face_loops(SELF))));
   WR2: SIZEOF(QUERY(temp <* bounds |
            'PART_204_BREP_PRODUCT_SCHEMA.FACE_OUTER_BOUND' IN
                                                   TYPEOF(temp))) <= 1;
 END_ENTITY;

 ENTITY face_surface
   SUBTYPE OF(face,geometric_representation_item);
   face_geometry :  surface;
   same_sense    :  BOOLEAN;
 END_ENTITY;
```

```
 ENTITY oriented_face
   SUBTYPE OF (face);
   face_element : face;
   orientation  : BOOLEAN;
 DERIVE
   bounds : SET[1:?] OF face_bound
          := conditional_reverse(SELF.orientation,SELF.face_element.bounds);
 WHERE
   WR1: NOT ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_FACE'
                              IN TYPEOF (SELF.face_element));
 END_ENTITY;

ENTITY advanced_face
  SUBTYPE OF (face_surface);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
  WR1 : aic_functionality =
             aic_topology_bounded_surface_functionality_definition;
  WR2 : SIZEOF (['AIC_TOP_BND_SURF.ELEMENTARY_SURFACE',
         'AIC_TOP_BND_SURF.B_SPLINE_SURFACE',
         'AIC_TOP_BND_SURF.SWEPT_SURFACE'] *
           TYPEOF(face_geometry)) = 1;
  WR3 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
          'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
         NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
             NOT('AIC_TOP_BND_SURF.EDGE_CURVE' IN
             TYPEOF(oe.edge_element)))) = 0))) = 0;
  WR4 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
          'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
           NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
           NOT (SIZEOF (['AIC_TOP_BND_SURF.LINE',
                        'AIC_TOP_BND_SURF.CONIC',
                        'AIC_TOP_BND_SURF.POLYLINE',
                        'AIC_TOP_BND_SURF.B_SPLINE_CURVE'] *
             TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
               )) = 0))) = 0;
  WR5 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
          'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
           NOT(SIZEOF(QUERY (oe <* elp_fbnds.bound\path.edge_list |
            NOT(('AIC_TOP_BND_SURF.VERTEX_POINT' IN TYPEOF(oe.edge_start)) AND
                 ('AIC_TOP_BND_SURF.VERTEX_POINT' IN TYPEOF(oe.edge_end))
             ))) = 0))) = 0;
  WR6 : SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* bounds |
          'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
             NOT('AIC_TOP_BND_SURF.ORIENTED_PATH' IN
              TYPEOF(elp_fbnds.bound)))) = 0;
  WR7 : NOT (('AIC_TOP_BND_SURF.SWEPT_SURFACE' IN TYPEOF(face_geometry)) AND
            (NOT SIZEOF(['AIC_TOP_BND_SURF.LINE',
                        'AIC_TOP_BND_SURF.CONIC',
                        'AIC_TOP_BND_SURF.POLYLINE',
                        'AIC_TOP_BND_SURF.B_SPLINE_CURVE'] *
                 TYPEOF(face_geometry\swept_surface.swept_curve)) = 1));
```

```
  WR8 : SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* bounds |
           'AIC_TOP_BND_SURF.VERTEX_LOOP' IN TYPEOF(bnds.bound)) |
             NOT(('AIC_TOP_BND_SURF.VERTEX_POINT' IN
                 TYPEOF(vlp_fbnds.bound\vertex_loop.loop_vertex)) AND
                  ('AIC_TOP_BND_SURF.CARTESIAN_POINT' IN
   TYPEOF(vlp_fbnds.bound\vertex_loop.loop_vertex\vertex_point.vertex_geometry))
              ))) = 0;
  WR9 : SIZEOF (QUERY (bnd <* bounds |
         NOT (SIZEOF(['AIC_TOP_BND_SURF.EDGE_LOOP',
         'AIC_TOP_BND_SURF.VERTEX_LOOP'] * TYPEOF(bnd.bound)) = 1))) = 0;
  END_ENTITY;



 ENTITY connected_face_set
   SUPERTYPE OF (ONEOF (closed_shell, open_shell))
   SUBTYPE OF (topological_representation_item);
   cfs_faces : SET [1:?] OF face;
 END_ENTITY;

 ENTITY closed_shell
   SUBTYPE OF (connected_face_set);
 END_ENTITY;

 ENTITY oriented_closed_shell
   SUBTYPE OF (closed_shell);
   closed_shell_element : closed_shell;
   orientation          : BOOLEAN;
 DERIVE
   cfs_faces : SET [1:?] OF face
             := conditional_reverse(SELF.orientation,
                                    SELF.closed_shell_element.cfs_faces);
 WHERE
   WR1: NOT ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL'
               IN TYPEOF (SELF.closed_shell_element));
 END_ENTITY;

ENTITY solid_model
  SUPERTYPE OF (ONEOF(manifold_solid_brep))
  SUBTYPE OF (geometric_representation_item);
END_ENTITY;

ENTITY manifold_solid_brep
  SUPERTYPE OF (facetted_brep ANDOR brep_with_voids)
  SUBTYPE OF (solid_model);
  outer : closed_shell;
END_ENTITY;


ENTITY brep_with_voids
  SUBTYPE OF (manifold_solid_brep);
  voids : SET [1:?] OF oriented_closed_shell;
```

```
END_ENTITY;

ENTITY facetted_brep
  SUBTYPE OF (manifold_solid_brep);
END_ENTITY;

ENTITY representation_item;
WHERE
  WR1: SIZEOF(using_representations(SELF)) +
       SIZEOF(using_definitional_representation_items(SELF)) > 0;
END_ENTITY;

ENTITY representation;
  items            : SET[1:?] OF representation_item;
  context_of_items : representation_context;
END_ENTITY;

ENTITY shape_representation
  SUBTYPE OF (representation);
END_ENTITY;

ENTITY representation_context;
  context_identifier : identifier;
  context_type       : text;
INVERSE
  contexted_representations : SET OF representation
    FOR context_of_items;
  contexted_definitional_representation_items : SET OF
    definitional_representation_item FOR context_of_items;
WHERE
  WR1: SIZEOF(SELF.contexted_representations) +
       SIZEOF(SELF.contexted_definitional_representation_items) > 0;
END_ENTITY;

ENTITY geometric_representation_context
  SUBTYPE OF (representation_context);
  coordinate_space_dimension : dimension_count;
END_ENTITY;

ENTITY representation_map;
  mapping_origin        : representation_item;
  mapped_representation : representation;
INVERSE
  map_usage : SET[1:?] OF mapped_item FOR mapping_source;
WHERE
  WR1: item_in_context(SELF.mapping_origin,
       SELF.mapped_representation.context_of_items);
END_ENTITY;

ENTITY mapped_item
  SUBTYPE OF (representation_item);
  mapping_source : representation_map;
```

```
  mapping_target : representation_item;
WHERE
  WR1: acyclic_mapped_representation(using_representations(SELF), [SELF]);
END_ENTITY;

ENTITY representation_relationship;
  rep_1 : representation;
  rep_2 : representation;
END_ENTITY;


ENTITY representation_relationship_with_transformation
  SUBTYPE OF (representation_relationship);
  transformation_operator : transformation;
WHERE
  WR1: SELF\representation_relationship.rep_1.context_of_items <>
       SELF\representation_relationship.rep_2.context_of_items;
END_ENTITY;



 ENTITY advanced_brep_shape_representation
   SUBTYPE OF (shape_representation);
     aic_functionality : LIST [2:2] OF STRING;
   WHERE
     WR1 : aic_functionality = aic_advanced_brep_functionality_definition;
     WR2 : SIZEOF (QUERY (it <* SELF.items |
            NOT (SIZEOF (['AIC_ADV_BREP.MANIFOLD_SOLID_BREP',
                  'AIC_ADV_BREP.FACETTED_BREP', 'AIC_ADV_BREP.MAPPED_ITEM',
                  'AIC_ADV_BREP.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1))) = 0;
     WR3 : SIZEOF (QUERY (it <* SELF.items |
           SIZEOF(['AIC_ADV_BREP.MANIFOLD_SOLID_BREP',
           'AIC_ADV_BREP.MAPPED_ITEM'] * TYPEOF(it)) =1 )) > 0;
     WR4 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
             'AIC_ADV_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
             NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_ADV_BREP') |
              NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
                NOT ('AIC_ADV_BREP.ADVANCED_FACE' IN TYPEOF(fcs)))) = 0
                   ))) = 0
                     ))) = 0;
     WR5 : SIZEOF (QUERY (msb <* QUERY (it <* items |
            'AIC_ADV_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
            'AIC_ADV_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF (msb.outer))) = 0;
     WR6 : SIZEOF (QUERY (brv <* QUERY (it <* items |
            'AIC_ADV_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) |
            NOT (SIZEOF (QUERY (csh <* brv.voids |
            NOT (('AIC_ADV_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF (csh)) AND
             (NOT (csh\oriented_closed_shell.orientation))))) = 0))) = 0;
     WR7 : SIZEOF (QUERY (mi <*  QUERY (it <* items |
                   'AIC_ADV_BREP.MAPPED_ITEM' IN TYPEOF(it)) |
            NOT ('AIC_ADV_BREP.ADVANCED_BREP_SHAPE_REPRESENTATION' IN
               TYPEOF(mi.mapping_source.mapped_representation)))) = 0;
 END_ENTITY;
```

```
ENTITY elementary_brep_shape_representation
SUBTYPE OF (shape_representation);
  aic_functionality : LIST [2:2} OF STRING;
WHERE
  WR1 : aic_functionality =
              aic_elementary_brep_functionality_definition;
  WR2 : SIZEOF (QUERY (it <* SELF.items |
         NOT (SIZEOF (['AIC_EL_BREP.MANIFOLD_SOLID_BREP',
              'AIC_EL_BREP.FACETTED_BREP', 'AIC_EL_BREP.MAPPED_ITEM',
              'AIC_EL_BREP.AXIS2_PLACEMENT_3D'] * TYPEOF(it)) = 1))) = 0;
  WR3 : SIZEOF (QUERY (it <* SELF.items |
        SIZEOF(['AIC_EL_BREP.MANIFOLD_SOLID_BREP',
        'AIC_EL_BREP.MAPPED_ITEM'] * TYPEOF(it)) =1 )) > 0;
  WR4 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
          'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
           NOT (SIZEOF (QUERY(fcs <* csh.cfs_faces |
             NOT('AIC_EL_BREP.FACE_SURFACE' IN TYPEOF(fcs)))) = 0
                ))) = 0
                  ))) = 0;
  WR5 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
          'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
           NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
             NOT(('AIC_EL_BREP.ELEMENTARY_SURFACE' IN
                  TYPEOF(fcs\face_surface.face_geometry))
            ))) = 0
                ))) = 0
                  ))) = 0;
  WR6 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
          'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
            NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
              NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
        'AIC_EL_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
               NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                 NOT('AIC_EL_BREP.EDGE_CURVE' IN
           TYPEOF(oe.edge_element)))) = 0
                  ))) = 0
                ))) = 0
              ))) = 0
            ))) = 0;
  WR7 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
          'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
          NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
            NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
              NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
        'AIC_EL_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
                NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
                  NOT (SIZEOF (['AIC_EL_BREP.LINE',
                        'AIC_EL_BREP.CONIC',
```

```
                      'AIC_EL_BREP.POLYLINE'] *
          TYPEOF(oe.edge_element\edge_curve.edge_geometry)) = 1 )
             )) = 0
             ))) = 0
               ))) = 0
                 ))) = 0
                  ))) = 0;
WR8 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
        NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
          NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.fcs_faces |
            NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
      'AIC_EL_BREP.EDGE_LOOP' IN TYPEOF(bnds.bound)) |
              NOT (SIZEOF (QUERY (oe <* elp_fbnds.bound\path.edge_list |
         NOT(('AIC_EL_BREP.VERTEX_POINT' IN TYPEOF(oe.edge_start)) AND
             ('AIC_EL_BREP.VERTEX_POINT' IN TYPEOF(oe.edge_end))
         ))) = 0
           ))) = 0
            ))) = 0
              ))) = 0
                ))) = 0;
WR9 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
        NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
          NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
            NOT (SIZEOF(QUERY (elp_fbnds <* QUERY (bnds <* fcs.bounds |
         'AIC_EL_BREP.ORIENTED_PATH' IN TYPEOF
           (elp_bnds.bound)))) = 0
            ))) = 0
              ))) = 0
                ))) = 0;
WR10 : SIZEOF (QUERY (msb <* QUERY (it <* items |
       'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
       'AIC_EL_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF (msb.outer))) = 0;
WR11 : SIZEOF (QUERY (brv <* QUERY (it <* items |
       'AIC_EL_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) |
       NOT (SIZEOF (QUERY (csh <* brv.voids |
       NOT (('AIC_EL_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF (csh)) AND
        (NOT (csh\oriented_closed_shell.orientation))))) = 0))) = 0;
WR12 : SIZEOF (QUERY (mi <*  QUERY (it <* items |
              'AIC_EL_BREP.MAPPED_ITEM' IN TYPEOF(it)) |
       NOT ('AIC_EL_BREP.ELEMENTARY_BREP_SHAPE_REPRESENTATION' IN
          TYPEOF(mi.mapping_source.mapped_representation)))) = 0;
WR13 : SIZEOF (QUERY (msb <* QUERY (it <* SELF.items |
        'AIC_EL_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
        NOT (SIZEOF (QUERY (csh <* msb_shells(msb, 'AIC_EL_BREP') |
          NOT (SIZEOF (QUERY(fcs <* csh\connected_face_set.cfs_faces |
            NOT (SIZEOF(QUERY (vlp_fbnds <* QUERY (bnds <* fcs.bounds |
      'AIC_EL_BREP.VERTEX_LOOP' IN TYPEOF(bnds.bound)) |
        NOT(('AIC_EL_BREP.VERTEX_POINT' IN
            TYPEOF(vlp_fbnds.bounds.loop_vertex)) AND
             ('AIC_EL_BREP.CARTESIAN_POINT' IN
```

```
                  TYPEOF(vlp_fbnds.bounds.loop_vertex.vertex_geometry))
           ))) = 0))) = 0))) = 0))) =0;
END_ENTITY;


ENTITY facetted_brep_shape_representation
  SUBTYPE OF (shape_representation);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
  WR1 : aic_functionality = aic_facetted_brep_functionality_definition;
  WR2 : SIZEOF (QUERY (it <* items |
         NOT (SIZEOF(['AIC_FAC_BREP.FACETTED_BREP',
           'AIC_FAC_BREP.MAPPED_ITEM',
           'AIC_FAC_BREP.AXIS2_PLACEMENT_3D'] *
            TYPEOF(it)) = 1))) = 0;
   WR3 : SIZEOF (QUERY (it <* items |
         SIZEOF(['AIC_FAC_BREP.FACETTED_BREP',
         'AIC_FAC_BREP.MAPPED_ITEM'] * TYPEOF(it)) = 1)) > 0;
   WR4 : SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
         'AIC_FAC_BREP.FACETTED_BREP' IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep, 'AIC_FAC_BREP') |
         NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
         NOT (('AIC_FAC_BREP.FACE_SURFACE' IN TYPEOF (fcs)) AND
         (('AIC_FAC_BREP.PLANE' IN TYPEOF (fcs.face_geometry)) AND
         ('AIC_FAC_BREP.CARTESIAN_POINT' IN TYPEOF (
          fcs.face_geometry\elementary_surface.position.location)))
             ))) = 0))) = 0))) = 0;
   WR5 : SIZEOF (QUERY (fbrep <* QUERY ( it <* items |
         'AIC_FAC_BREP.FACETTED_BREP' IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* msb_shells(fbrep, 'AIC_FAC_BREP') |
         NOT (SIZEOF (QUERY (fcs <* csh.cfs_faces |
         NOT (SIZEOF (QUERY (bnds <* fcs.bounds |
          'AIC_FAC_BREP.FACE_OUTER_BOUND' IN TYPEOF(bnds)))
             = 1))) = 0))) = 0))) = 0;
   WR6 : SIZEOF (QUERY (msb <* QUERY (it <* items |
         'AIC_FAC_BREP.MANIFOLD_SOLID_BREP' IN TYPEOF(it)) |
         'AIC_FAC_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF (msb.outer))) = 0;
   WR7 : SIZEOF (QUERY (brv <* QUERY (it <* items |
         'AIC_FAC_BREP.BREP_WITH_VOIDS' IN TYPEOF(it)) |
         NOT (SIZEOF (QUERY (csh <* brv.voids |
         NOT (('AIC_FAC_BREP.ORIENTED_CLOSED_SHELL' IN TYPEOF (csh)) AND
          (NOT (csh\oriented_closed_shell.orientation))))) = 0))) = 0;
   WR8 : SIZEOF (QUERY (mi <*  QUERY (it <* items |
                'AIC_FAC_BREP.MAPPED_ITEM' IN TYPEOF(it)) |
         NOT ('AIC_FAC_BREP.FACETTED_BREP_SHAPE_REPRESENTATION' IN
            TYPEOF(mi.mapping_source.mapped_representation)))) = 0;
END_ENTITY;



ENTITY oriented_path
  SUBTYPE OF (path);
  path_element : path;
```

```
   orientation  : BOOLEAN;
 DERIVE
   edge_list    : LIST [1:?] OF UNIQUE oriented_edge
                := conditional_reverse(SELF.orientation,
                                        SELF.path_element.edge_list);
 WHERE
   WR1: NOT ('ORIENTED_PATH' IN TYPEOF (SELF.path_element));
 END_ENTITY;

 ENTITY open_shell
   SUBTYPE OF (connected_face_set);
 END_ENTITY;

 ENTITY oriented_open_shell
   SUBTYPE OF (open_shell);
   open_shell_element : open_shell;
   orientation        : BOOLEAN;
 DERIVE
   cfs_faces : SET [1:?] OF face
             := conditional_reverse(SELF.orientation,
                                    SELF.open_shell_element.cfs_faces);
 WHERE
   WR1: NOT ('ORIENTED_OPEN_SHELL' IN TYPEOF (SELF.open_shell_element));
 END_ENTITY;

 ENTITY definitional_representation_item;
   items           : SET [1:?] of representation_item;
   context_of_items : parametric_representation_context;
 WHERE
    WR1: SIZEOF(USEDIN(SELF,'')) > 0;
 END_ENTITY;

 ENTITY parametric_representation_context
   SUBTYPE OF (geometric_representation_context);
 END_ENTITY;

(* entities from mechanical design context AIC *)

ENTITY application_context;
  status                                  : label;
  application_interpreted_model_schema_name : label;
  application_protocol_year               : year_number;
  application                             : text;
INVERSE
  context_elements : SET [1:?] OF  application_context_element
                              FOR frame_of_reference;
END_ENTITY;

ENTITY application_context_element
  SUPERTYPE OF (ONEOF (product_context,
                       product_definition_context,
                       product_concept_context,
```

```
                       library_context));
  name                 : label;
  frame_of_reference   : application_context;
END_ENTITY;

ENTITY assembly_component_usage
  SUBTYPE OF (product_definition_usage);
  reference_designator : OPTIONAL identifier;
 END_ENTITY;

ENTITY design_context
  SUBTYPE OF (product_definition_context)
    aic_function : LIST [2:2] OF STRING;
  WHERE
    WR1 : aic_function =
              mechanical_design_context_aic_function_definition;
    WR2 : SELF.life_cycle_stage = 'design';
 END_ENTITY;

ENTITY mechanical_context
  SUBTYPE OF (product_context)
    aic_function : LIST [2:2] OF STRING;
  WHERE
    WR1 : aic_function =
              mechanical_design_context_aic_function_definition;
    WR2 : SELF.discipline_type = 'mechanical';
 END_ENTITY;

ENTITY product;
  id                 : identifier;
  name               : label;
  description        : text;
  frame_of_reference : product_context;
UNIQUE
  UR1: id;
END_ENTITY;

ENTITY product_context
  SUBTYPE OF (application_context_element);
  discipline_type : label;
END_ENTITY;


ENTITY product_definition_context
  SUBTYPE OF (application_context_element);
  life_cycle_stage : label;
END_ENTITY;

ENTITY product_definition_relationship;
  id                          : identifier;
  name                        : label;
  description                 : text;
  relating_product_definition : product_definition;
```

```
  related_product_definition  : product_definition;
END_ENTITY;

ENTITY product_definition_usage
SUPERTYPE OF (ONEOF (make_from_usage_option,
   assembly_component_usage))
  SUBTYPE OF (product_definition_relationship);

UNIQUE
  UR1:id, relating_product_definition, related_product_definition;
WHERE
  WR1:  acyclic_product_definition_relationship
         (SELF,
         [SELF\product_definition_relationship.related_product_definition],
         'PRODUCT_STRUCTURE.PRODUCT_DEFINITION_USAGE.' +
         'RELATED_PRODUCT_DEFINITION');
END_ENTITY;

ENTITY product_version;
  id          : identifier;
  description : text;
  of_product  : product;
UNIQUE
  UR1: id, of_product;
END_ENTITY;



(*
Measure and unit entities and functions from Part 41 - to be verified
*)

ENTITY measure_with_unit;
  value_component : measure_value;
  unit_component  : unit;
WHERE
  WR1: valid_units (SELF);
END_ENTITY;

ENTITY length_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: TYPEOF (SELF.unit_component)  = 'MEASURE_SCHEMA.LENGTH_UNIT';
  WR2: TYPEOF (SELF.value_component) IN
       ['MEASURE_SCHEMA.LENGTH_MEASURE', 'MEASURE_SCHEMA.POSITIVE_LENGTH_MEASURE'];
END_ENTITY;

ENTITY plane_angle_measure_with_unit
  SUBTYPE OF (measure_with_unit);
WHERE
  WR1: TYPEOF (SELF.unit_component) = 'MEASURE_SCHEMA.PLANE_ANGLE_UNIT';
  WR2: TYPEOF (SELF.value_component) IN
       ['MEASURE_SCHEMA.PLANE_ANGLE_MEASURE',
```

```
        'MEASURE_SCHEMA.POSITIVE_PLANE_ANGLE_MEASURE'];
END_ENTITY;


ENTITY global_unit_assigned_context
  SUBTYPE OF (representation_context);
  units : SET [1:?] OF unit;
END_ENTITY;

ENTITY named_unit
  SUPERTYPE OF (ONEOF (si_unit, conversion_based_unit));
  dimensions : dimensional_exponents;
END_ENTITY;

ENTITY si_unit
  SUBTYPE OF (named_unit);
  prefix     : OPTIONAL si_prefix;
  name       : si_unit_name;
DERIVE
  dimensions : dimensional_exponents := dimensions_for_si_unit (SELF.name);
END_ENTITY;

ENTITY conversion_based_unit
  SUBTYPE OF (named_unit);
  name              : label;
  conversion_factor : measure_with_unit;
END_ENTITY;

ENTITY length_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF.dimensions.length_exponent = 1.0)                    AND
       (SELF.dimensions.mass_exponent = 0.0)                      AND
       (SELF.dimensions.time_exponent = 0.0)                      AND
       (SELF.dimensions.electric_current_exponent = 0.0)          AND
       (SELF.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF.dimensions.amount_of_substance_exponent = 0.0)       AND
       (SELF.dimensions.luminous_intensity_exponent = 0.0);
END_ENTITY;

ENTITY plane_angle_unit
  SUBTYPE OF (named_unit);
WHERE
  WR1: (SELF.dimensions.length_exponent                    = 0.0) AND
       (SELF.dimensions.mass_exponent                      = 0.0) AND
       (SELF.dimensions.time_exponent                      = 0.0) AND
       (SELF.dimensions.electric_current_exponent          = 0.0) AND
       (SELF.dimensions.thermodynamic_temperature_exponent = 0.0) AND
       (SELF.dimensions.amount_of_substance_exponent       = 0.0) AND
       (SELF.dimensions.luminous_intensity_exponent        = 0.0);
END_ENTITY;
```

```
 (* Entities from presentation AIC - to be verified and completed*)


ENTITY mechanical_design_presentation_area
  SUBTYPE OF (presentation_area);
  aic_functionality : LIST [2:2] OF STRING;
WHERE
    WR1 : aic_functionality =
           aic_mechanical_design_presentation_functionality_definition;

    WR2 : -- legal_representations FOR (camera_usage);
        -- get for all presentation_areas their presentation_views
          SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
          'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
          'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
          (mi1\mapped_item.mapping_source.mapped_representation)) |
        -- get for all presentation_views their
        -- product_data_representation_views
          NOT (SIZEOF (QUERY (pdrv <* QUERY (mi2 <* QUERY (it2 <*
          pv.items |
          'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
          'AIC_MECH_DSGN_PRES.PRODUCT_DATA_REPRESENTATION_VIEW' IN TYPEOF
          (mi2\mapped_item.mapping_source.mapped_representation)) |
        -- a product_data_representation_view has exactly one item in its list
        -- of items and this is a camera_image;
        -- get the camera_usage of this camera_image
          NOT (SIZEOF (QUERY (cu <* QUERY (it3 <* pdrv.items |
          'AIC_MECH_DSGN_PRES.CAMERA_USAGE' IN TYPEOF
          (it3\mapped_item.mapping_source)) |
        -- apply the test that a camera_usage shall have a
        -- shape_representation or a
        -- mechanical_design_presentation_representation as its
        -- mapped_representation
          NOT (SIZEOF (['AIC_MECH_DSGN_PRES.SHAPE_REPRESENTATION',
          'AIC_MECH_DSGN_PRES.MECHANICAL_DESIGN_PRESENTATION_REPRESENTATION']
          * TYPEOF (cu\representation_map.mapped_representation)) = 1 )))
          = 0 ))) = 0 ))) = 0;

    WR3 : -- mandatory_curve_style FOR (curved styled_item);
        -- get for all presentation_areas their presentation_views
          SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
          'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
          'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
          (mi1\mapped_item.mapping_source.mapped_representation)) |
        -- get for all presentation_views their
        -- product_data_representation_views
          NOT (SIZEOF (QUERY (pdrv <* QUERY (mi2 <* QUERY (it2 <*
          pv.items |
          'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
          'AIC_MECH_DSGN_PRES.PRODUCT_DATA_REPRESENTATION_VIEW' IN TYPEOF
          (mi2\mapped_item.mapping_source.mapped_representation)) |
```

```
    -- a product_data_representation_view has exactly one item in its list
    -- of items and this is a camera_image; get the camera_usage of this
    -- camera_image and the mapped_representation of type
    -- shape_representation
      NOT (SIZEOF (QUERY (sr <* QUERY (cu <* QUERY (it3 <* pdrv.items |
      'AIC_MECH_DSGN_PRES.CAMERA_USAGE' IN TYPEOF
      (it3\mapped_item.mapping_source)) |
      'AIC_MECH_DSGN_PRES.SHAPE_REPRESENTATION'
      IN TYPEOF (cu\representation_map.mapped_representation)) |
    -- get all items of the mapped_representation of the camera_usage;
    -- filter out those of type styled_items and out of these the curves
      NOT (SIZEOF (QUERY (cv <* QUERY (si <* QUERY (it4 <* sr.items |
      'AIC_MECH_DSGN_PRES.STYLED_ITEM' IN TYPEOF (it4)) |
      'GEOMETRY_SCHEMA.CURVE' IN TYPEOF (si.item)) |
    -- apply the restriction that a curve in styled_item needs a
    -- curve_style
      NOT (SIZEOF (QUERY (psa <* si.styles | SIZEOF (psa.styles) = 1 AND
      'AIC_MECH_DSGN_PRES.CURVE_STYLE' IN TYPEOF (psa.styles[1])))
      = 0 ))) = 0 ))) = 0 ))) = 0))) = 0;

WR4 : -- mandatory_point_style FOR (pointed styled_item);
    -- get for all presentation_areas their presentation_views
      SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
      'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
      'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
      (mi1\mapped_item.mapping_source.mapped_representation)) |
    -- get for all presentation_views their
    -- product_data_representation_views
      NOT (SIZEOF (QUERY (pdrv <* QUERY (mi2 <* QUERY (it2 <*
      pv.items |
      'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
      'AIC_MECH_DSGN_PRES.PRODUCT_DATA_REPRESENTATION_VIEW' IN TYPEOF
      (mi2\mapped_item.mapping_source.mapped_representation)) |
    -- a product_data_representation_view has exactly one item in its list
    -- of items and this is a camera_image; get the camera_usage of this
    -- camera_image and the mapped_representation of type
    -- shape_representation
      NOT (SIZEOF (QUERY (sr <* QUERY (cu <* QUERY (it3 <* pdrv.items |
      'AIC_MECH_DSGN_PRES.CAMERA_USAGE' IN TYPEOF
      (it3\mapped_item.mapping_source)) |
      'AIC_MECH_DSGN_PRES.SHAPE_REPRESENTATION'
      IN TYPEOF (cu\representation_map.mapped_representation)) |
    -- get all items of the mapped_representation of the camera_usage;
    -- filter out those of type styled_items and out of these the points
      NOT (SIZEOF (QUERY (pnt <* QUERY (si <* QUERY (it4 <* sr.items |
      'AIC_MECH_DSGN_PRES.STYLED_ITEM' IN TYPEOF (it4)) |
      'GEOMETRY_SCHEMA.POINT' IN TYPEOF (si.item)) |
    -- apply the restriction that a point in styled_item needs a
    -- point_style
      NOT (SIZEOF (QUERY (psa <* si.styles | SIZEOF (psa.styles) = 1 AND
      'AIC_MECH_DSGN_PRES.POINT_STYLE' IN TYPEOF (psa.styles[1])))
      = 0 ))) = 0 ))) = 0 ))) = 0))) = 0;
```

```
WR5 : -- predefined_text_style FOR (styled annotation_text);
    -- get for all presentation_areas their presentation_views
      SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
      'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
      'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
      (mi1\mapped_item.mapping_source.mapped_representation)) |
    -- get for all presentation_views their
    -- view_dependent_annotation_representations
      NOT (SIZEOF (QUERY (vdar <* QUERY (mi2 <* QUERY (it2 <*
      pv.items |
      'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
      'AIC_MECH_DSGN_PRES.VIEW_DEPENDENT_ANNOTATION_REPRESENTATION'
      IN TYPEOF
      (mi2\mapped_item.mapping_source.mapped_representation)) |
    -- get for all view_dependent_annotation_representations their
    -- annotation_text_occurrences
      NOT (SIZEOF (QUERY (atc <* QUERY (it3 <* vdar.items |
      'AIC_MECH_DSGN_PRES.ANNOTATION_TEXT_OCCURRENCE' IN TYPEOF (it3)) |
    -- apply the test that all annotation_text_occurrence shall reference
    -- an annotation_text and shall have only one style that is the
    -- mechanical_design_pre_defined_text_style
      NOT ('AIC_MECH_DSGN_PRES.ANNOTATION_TEXT' IN TYPEOF (atc.item)) AND
      NOT (SIZEOF (QUERY (psa <* atc.styles |
      NOT (SIZEOF (psa.styles) = 1) AND
      NOT ('AIC_MECH_DSGN_PRES.MECHANICAL_DESIGN_PRE_DEFINED_TEXT_STYLE'
      IN TYPEOF (psa.styles[1])))) = 0 ))) = 0 ))) = 0 ))) = 0;

WR6 : -- mandatory_mech_des_pre_text_font FOR (annotation_text);
    -- get for all presentation_areas their presentation_views
      SIZEOF (QUERY (pv <* QUERY (mi1 <* QUERY (it1 <* items |
      'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it1)) |
      'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW' IN TYPEOF
      (mi1\mapped_item.mapping_source.mapped_representation)) |
    -- get for all presentation_views their
    -- view_dependent_annotation_representations
      NOT (SIZEOF (QUERY (vdar <* QUERY (mi2 <* QUERY (it2 <*
      pv.items |
      'AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF (it2)) |
      'AIC_MECH_DSGN_PRES.VIEW_DEPENDENT_ANNOTATION_REPRESENTATION'
      IN TYPEOF
      (mi2\mapped_item.mapping_source.mapped_representation)) |
    -- get for all view_dependent_annotation_representations their
    -- annotation_text_occurrences
      NOT (SIZEOF (QUERY (atc <* QUERY (it3 <* vdar.items |
      'AIC_MECH_DSGN_PRES.ANNOTATION_TEXT_OCCURRENCE' IN TYPEOF (it3)) |
    -- apply the restriction that each annotation_text shall have the
    -- mechanical_design_pre_defined_text_font; that the item of an
    -- annotation_text_occurrence is an annotation_text has been checked
    -- above; use function to cover the recursion that an annotation_text
    -- may be in the list of strings of a text_literal_mapped_item
      NOT (SIZEOF (QUERY (tlmi <*
```

```
          mdp_get_text_literal_mapped_item_in_annotation_text
          (atc.item, 'AIC_MECH_DSGN_PRES') |
          'AIC_MECH_DSGN_PRES.MECHANICAL_DESIGN_PRE_DEFINED_TEXT_FONT'
          IN TYPEOF (tlmi\mapped_item.mapped_representation.font)))
          = 0 ))) = 0 ))) = 0 ))) = 0;

END_ENTITY;

ENTITY camera_model
SUPERTYPE OF (ONEOF(camera_model_d3))
SUBTYPE OF(geometric_representation_item,
           representation_item_without_style);
WHERE
   WR1: SIZEOF(USEDIN(SELF,
               'AIC_MECH_DSGN_PRES.REPRESENTATION_RELATIONSHIP_'+
               'WITH_TRANSFORMATION.TRANSFORMATION_OPERATOR' +
                \ITEM_DEFINED_TRANSFORMATION.TRANSFORM_ITEM_1') +
               USEDIN(SELF, 'AIC_MECH_DSGN_PRES.' +
              'CAMERA_USAGE.PROJECTION')) > 0;
END_ENTITY;

ENTITY camera_model_d3
SUBTYPE OF(camera_model);
   view_reference_system : axis2_placement_3d;
   perspective_of_volume : view_volume;
WHERE
   WR1: (dot_product(SELF.view_reference_system.p[3],
           SELF.perspective_of_volume.view_window.placement.p[3]) = 1.0)
         AND
        (SELF.view_reference_system.location.coordinates[3] =
         SELF.perspective_of_volume.view_window.placement.location.
         coordinates[3]);
   WR2: SELF\geometric_representation_item.dim = 3;
END_ENTITY;

ENTITY camera_image
SUBTYPE OF(geometric_representation_item, mapped_item);
DERIVE
   source   : camera_usage := SELF\mapped_item.mapping_source;
   viewport : planar_box := SELF\mapped_item.mapping_target;
END_ENTITY;

ENTITY camera_usage
SUBTYPE OF(representation_map);
DERIVE
   projection : camera_model := SELF\representation_map.mapping_origin;
WHERE
   WR1: NOT('AIC_MECH_DSGN_PRES.PRESENTATION_REPRESENTATION'
        IN TYPEOF(SELF\representation_map..mapped_representation));
END_ENTITY;

ENTITY view_volume;
```

```
   perspective_type           : central_or_parallel;
   projection_point           : cartesian_point;
   view_plane_distance        : length_measure;
   front_plane_distance       : length_measure;
   front_plane_clipping       : BOOLEAN;
   back_plane_distance        : length_measure;
   back_plane_clipping        : BOOLEAN;
   view_volume_sides_clipping : BOOLEAN;
   view_window                : planar_box;
WHERE
   WR1: SELF.front_plane_distance < SELF.back_plane_distance;
   WR2: SELF.view_window.placement.location.coordinates[3]
          = SELF.view_plane_distance;
   WR3: SELF.projection_point.coordinates[3] <> SELF.view_plane_distance;
END_ENTITY;

ENTITY representation_item_without_style
SUBTYPE OF (representation_item);
WHERE
   WR1: SIZEOF(USEDIN(SELF,'AIC_MECH_DSGN_PRES.'+
                         'STYLED_ITEM')) = 0;
END_ENTITY;

ENTITY pre_defined_item;
  name : label;
END_ENTITY;

ENTITY predefined_presentation_style
SUBTYPE OF (pre_defined_item);
END_ENTITY;

ENTITY mechanical_design_pre_defined_text_style
  SUBTYPE OF (pre_defined_presentation_style);
WHERE
  WR1: SELF.name IN ['mechanical design text style'];
END_ENTITY;

ENTITY curve_style_font;
   pattern_list : LIST [1:?] OF curve_style_font_pattern;
END_ENTITY;

ENTITY curve_style_font_pattern;
   visible_segment_length   : positive_length_measure;
   invisible_segment_length : positive_length_measure;
END_ENTITY;

ENTITY point_style;
   marker        : marker_select;
   marker_size   : size_select;
   marker_colour : colour;
END_ENTITY;
```

```
ENTITY curve_style;
   curve_font        : curve_font_or_scaled_curve_font_select;
   curve_width       : size_select;
   curve_colour      : colour;
END_ENTITY;

ENTITY surface_style_usage;
   side  : surface_side;
   style : surface_side_style_select;
END_ENTITY;

ENTITY surface_side_style;
   styles : SET [1:7] OF surface_style_element_select;
WHERE
   WR1: SIZEOF(QUERY( style1 <* SELF.styles |
               SIZEOF(QUERY( style2 <* SELF.styles - style1 |
                      TYPEOF(style1) = TYPEOF(style2)
               ))>0
        )) =0;
END_ENTITY;

ENTITY curve_style_rendering;
   rendering_method     : shading_curve_method;
   rendering_properties : surface_rendering_properties;
END_ENTITY;

ENTITY surface_style_boundary;
   style_of_boundary : curve_or_render;
END_ENTITY;

ENTITY surface_style_silhouette;
   style_of_silhouette : curve_or_render;
END_ENTITY;

ENTITY surface_style_segmentation_curve;
   style_of_segmentation_curve : curve_or_render;
END_ENTITY;

ENTITY surface_style_control_grid;
   style_of_control_grid : curve_or_render;
END_ENTITY;

ENTITY surface_style_parameter_line;
   style_of_parameter_lines : curve_or_render;
   direction_counts         : SET [1:2] OF direction_count_select;
WHERE
   WR1: (HIINDEX(SELF.direction_counts) = 1)
                      XOR
        (TYPEOF(SELF.direction_counts[1]) <>
                      TYPEOF(SELF.direction_counts[2]));
END_ENTITY;
```

```
ENTITY surface_style_rendering;
   rendering_method : shading_surface_method;
   surface_colour : colour;
END_ENTITY;

ENTITY surface_rendering_properties;
   rendered_colour : colour;
END_ENTITY;

ENTITY presentation_style_assignment;
   styles : SET [1:?] OF presentation_style_select;
WHERE
   WR1: SIZEOF(QUERY(style1 <* SELF.styles |
              SIZEOF(QUERY(style2 <* SELF.styles - style1    |
                  (
                   (TYPEOF(style1) = TYPEOF(style2))
                          AND
                   ((SIZEOF(['AIC_MECH_DSGN_PRES.'+
                              'SURFACE_STYLE_USAGE',
                              'AIC_MECH_DSGN_PRES.'+
                              'EXTERNALLY_DEFINED_STYLE'] *
                             TYPEOF(style1)) = 0)
                                       XOR
                    (('AIC_MECH_DSGN_PRES.SURFACE_STYLE_USAGE'
                      IN TYPEOF(style1))
                           AND
                     ((style1.side = style2.side)
                           OR
                      (style1.side = both)) )
                    )
                   )
              ))>0
        ))=0;
END_ENTITY;

ENTITY presentation_style_by_context
SUBTYPE OF(presentation_style_assignment);
   style_context : style_context_select;
END_ENTITY;


ENTITY styled_item
SUBTYPE OF (representation_item);
  styles :  SET [1:?] OF presentation_style_assignment;
  item   :  representation_item;
WHERE
  WR1:  (SIZEOF(SELF.styles) = 1)
              XOR
        (SIZEOF(QUERY(pres_style <* SELF.styles |
            NOT('AIC_MECH_DSGN_PRES.PRESENTATION_STYLE_BY_CONTEXT'
                 IN TYPEOF(pres_style))
          )) = 0);
END_ENTITY;
```

```
ENTITY representation_dependent_styled_item
SUBTYPE OF (styled_item);
  rep_using_item:  representation;
WHERE
   WR1: SELF.rep_using_item = using_representations(SELF\styled_item.item);
END_ENTITY;

ENTITY annotation_occurrence
SUPERTYPE OF(ONEOF(annotation_text_occurrence))
SUBTYPE OF(styled_item);
WHERE
   WR1: SIZEOF(QUERY(each <* USEDIN(SELF,'') |
            SIZEOF(TYPEOF(each) *
                   ['AIC_MECH_DSGN_PRES.'+
                    'AREA_DEPENDENT_ANNOTATION_REPRESENTATION',
                    'AIC_MECH_DSGN_PRES.'+
                    'VIEW_DEPENDENT_ANNOTATION_REPRESENTATION',
                    'AIC_MECH_DSGN_PRES.'+
                    'GEOMETRIC_VIEW_DEPENDENT_ANNOTATION_REPRESENTATION',
                    'AIC_MECH_DSGN_PRES.'+
                    'CURVE_STYLE_CURVE_PATTERN',
                    'AIC_MECH_DSGN_PRES.'+
                    'FILL_AREA_STYLE_TILE_CURVE_WITH_STYLE',
                    'AIC_MECH_DSGN_PRES.'+
                    'FILL_AREA_STYLE_COLOURED_REGION']) =0 ))=0;
   WR2:  NOT ( 'AIC_MECH_DSGN_PRES.' +
         'REPRESERNTATION_DEPENDENT_STYLED_ITEM' IN TYPEOF(SELF));
END_ENTITY;


ENTITY annotation_text_occurrence
SUBTYPE OF(annotation_occurrence);
WHERE
   WR1: 'AIC_MECH_DSGN_PRES.ANNOTATION_TEXT'
           IN TYPEOF(SELF\styled_item.item);
END_ENTITY;

ENTITY annotation_text
SUBTYPE OF(mapped_item, geometric_representation_item);
DERIVE
   text_source : annotation_text_map := SELF\mapped_item.mapping_source;
   placement   : axis2_placement     := SELF\mapped_item.mapping_target;
END_ENTITY;

ENTITY text_symbol
  SUBTYPE OF (symbol_representation);
END_ENTITY;

ENTITY symbol_representation
   SUBTYPE OF representation;
END_ENTITY;
```

```
ENTITY text_literal_symbol
SUBTYPE OF(text_symbol);
   literal   : presentable_text;
   origin    : axis2_placement;
   alignment : text_alignment;
   path      : text_path;
   font      : font_select;
DERIVE
   SELF\representation.items:
     SET [1:1] OF axis2_placement := [SELF.origin];
END_ENTITY;

ENTITY text_literal_mapped_item
  SUBTYPE OF(mapped_item);
WHERE
  WR1: 'AIC_MECH_DSGN_PRES.TEXT_LITERAL_SYMBOL' IN
         TYPEOF(SELF\mapped_item.mapping_source.mapped_representation);
  WR2: USEDIN (SELF, '' ) = USEDIN (SELF, 'AIC_MECH_DSGN_PRES' +
                               TEXT_STRING_REPRESENTATION.STRINGS');
END_ENTITY;


ENTITY predefined_text_font
SUBTYPE OF (pre_defined_item);
END_ENTITY;

ENTITY mechanical_design_pre_defined_text_font
  SUBTYPE OF (pre_defined_text_font);
WHERE
  WR1: SELF.name IN ['ISO 3098 font'];
END_ENTITY;

ENTITY text_string_representation
SUBTYPE OF(representation);
   strings   : SET [1:?] OF text_or_character;
   placement : axis2_placement;
DERIVE
  SELF\representation.items : SET[1:?] OF representation_item :=
                                SELF.strings + [SELF.placement];
WHERE
   WR1: SIZEOF(USEDIN(SELF, 'AIC_MECH_DSGN_PRES.'+
                         'ANNOTATION_TEXT_MAP.TEXT_STRING'))>0;
   WR2: SIZEOF (QUERY (str <* SELF.strings |
                NOT (('AIC_MECH_DSGN_PRES.' +
                      'TEXT_LITERAL_MAPPED_ITEM' IN TYPEOF (str))
                          AND
                     (str.mapping_target :=: SELF.placement)))) = 0;
END_ENTITY;

ENTITY colour;
END_ENTITY;
```

```
ENTITY colour_specification
  SUBTYPE OF (colour);
END_ENTITY;

ENTITY colour_rgb
SUBTYPE OF (colour_specification);
   red   : REAL;
   green : REAL;
   blue  : REAL;
WHERE
   WR1 : {0.0 <= red <= 1.0};
   WR2 : {0.0 <= green <= 1.0};
   WR3 : {0.0 <= blue <= 1.0};
END_ENTITY;

ENTITY annotation_text_map
SUBTYPE OF(representation_map);
DERIVE
   text_string : text_string_representation :=
                  SELF\representation_map.mapped_representation;
WHERE
   WR1: SIZEOF(USEDIN(SELF,
        'AIC_MECH_DSGN_PRES.ANNOTATION_TEXT.TEXT_SOURCE')) > 0;
   WR2: SELF\representation_map.mapping_origin :=: SELF.text_string.placement;
END_ENTITY;

ENTITY presentation_area
SUBTYPE OF (presentation_representation);
WHERE
   WR1: (SIZEOF(USEDIN(SELF, 'AIC_MECH_DSGN_PRES.'
              + 'PRESENTATION_REPRESENTATION_RELATIONSHIP.'
              + 'PARENT_REPRESENTATION')) > 0)
            OR
        (SIZEOF(QUERY( item <* SELF\representation.items |
           ('AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF(item))) ) > 0);
  WR2: ( ('AIC_MECH_DSGN_PRES.AREA_IN_SET' IN TYPEOF(SELF))
          OR
          SIZEOF(USEDIN(SELF, 'AIC_MECH_DSGN_PRES.'
                            +'PRESENTATION_SIZE.UNIT'))=1 );
  WR3: SIZEOF(QUERY( item <* SELF\representation.items |
            ('AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF(item))
                         AND
             (SIZEOF(['AIC_MECH_DSGN_PRES.PRESENTATION_AREA',
                      'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW',
                     'AIC_MECH_DSGN_PRES.'+
                     'AREA_DEPENDENT_ANNOTATION_REPRESENTATION'] *
         TYPEOF(item\mapped_item.mapping_source.mapped_representation))
          = 0) )) = 0;
END_ENTITY;

ENTITY background_colour
SUBTYPE OF(colour);
```

```
   presentation : area_or_view;
UNIQUE
   UR1:  presentation;
END_ENTITY;


ENTITY presentation_representation
   SUBTYPE OF (representation);
WHERE
   WR1: SELF\representation.context_of_items\geometric_representation_context.
               coordinate_space_dimension = 2;
   WR2: 'AIC_MECH_DSGN_PRES.GEOMETRIC_REPRESENTATION_CONTEXT'
         IN TYPEOF(SELF\representation.context_of_items);
END_ENTITY;

ENTITY mechanical_design_presentation_representation
   SUBTYPE OF (presentation_representation);
WHERE
  WR1 : SIZEOF (QUERY (it <* SELF.items |
        NOT ('AIC_MECH_DSGN_PRES.STYLED_ITEM' IN TYPEOF (it)))) = 0;
END_ENTITY;

ENTITY presentation_view
SUBTYPE OF(presentation_representation);
WHERE
   WR1: (SIZEOF(USEDIN(SELF, 'AIC_MECH_DSGN_PRES.'
                      + 'PRESENTATION_REPRESENTATION_RELATIONSHIP.'
                      + 'PARENT_REPRESENTATION')) > 0)
               OR
        (SIZEOF(QUERY( item <* SELF\representation.items |
                 ('AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF(item))
                   )) > 0);
   WR2: ( ('AIC_MECH_DSGN_PRES.AREA_IN_SET' IN TYPEOF(SELF))
             OR
          (SIZEOF(USEDIN(SELF, 'AIC_MECH_DSGN_PRES.'
                             + 'PRESENTATION_SIZE.UNIT')) = 1) );
   WR3: SIZEOF(QUERY( item <* SELF\representation.items |
            ('AIC_MECH_DSGN_PRES.MAPPED_ITEM' IN TYPEOF(item))
                         AND
            (SIZEOF(['AIC_MECH_DSGN_PRES.PRESENTATION_AREA',
                     'AIC_MECH_DSGN_PRES.PRESENTATION_VIEW',
                     ''AIC_MECH_DSGN_PRES.'+
                     'AREA_DEPENDENT_ANNOTATION_REPRESENTATION'] *
               TYPEOF(item\mapped_item.mapping_source.mapped_representation)
             ) = 0)
      )) = 0;
END_ENTITY;

ENTITY view_dependent_annotation_representation
SUBTYPE OF (presentation_representation);
WHERE
   WR1: SIZEOF(USEDIN(SELF, ''AIC_MECH_DSGN_PRES.'
               + 'PRESENTATION_REPRESENTATION_RELATIONSHIP.'
```

```
              + 'PARENT_REPRESENTATION')) = 0;
   WR2: SIZEOF(QUERY(item <* SELF.items |
              SIZEOF([''AIC_MECH_DSGN_PRES.'+
                      'ANNOTATION_OCCURRENCE',
                      ''AIC_MECH_DSGN_PRES.AXIS2_PLACEMENT'] * TYPEOF(item)
                    ) = 0
            )) = 0;
END_ENTITY;

ENTITY product_data_representation_view
SUBTYPE OF(presentation_representation);
WHERE
   WR1 : (SIZEOF(SELF\representation.items) = 1)
             AND
         (''AIC_MECH_DSGN_PRES.CAMERA_IMAGE'
          IN TYPEOF (SELF.items[1]));
END_ENTITY;

ENTITY product_data_representation_view_with_hlhsr
  SUBTYPE OF(product_data_representation_view);
  hidden_line_surface_removal : BOOLEAN;
END_ENTITY;

ENTITY planar_extent
SUBTYPE OF (geometric_representation_item);
   size_in_x : length_measure;
   size_in_y : length_measure;
END_ENTITY;

ENTITY planar_box
SUBTYPE OF (planar_extent, curve);
  placement:  axis2_placement;
END_ENTITY;

ENTITY presentation_layer_usage;
   assignment     : presentation_layer_assignment;
   representation : presentation_representation;
   visibility     : LOGICAL;
UNIQUE
   U1: assignment, representation;
END_ENTITY;

ENTITY presentation_layer_assignment;
   layer_value            : identifier;
   layered_representation : SET [1:?] OF layered_item;
END_ENTITY;

ENTITY presentation_size;
   unit : presentation_size_assigment_select;
   size : planar_box;
WHERE
   WR1: ( (''AIC_MECH_DSGN_PRES.PRESENTATION_REPRESENTATION'
```

```
            IN TYPEOF(SELF.unit))
               AND
          item_in_context(SELF.size,
                    SELF.unit\representation.context_of_items) )
             OR
        ( (''AIC_MECH_DSGN_PRES.SIZE_OF_AREA_IN_SET'
          IN TYPEOF(SELF.unit))
               AND
        (SIZEOF(QUERY( area <* SELF.unit\size_of_area_in_set.of_set.areas |
                 NOT item_in_context(SELF.size, area.context_of_items)
                 )) = 0)
        );
END_ENTITY;



RULE
advanced_or_elementary_or_facetted FOR(shape_representation);
WHERE
   WR1: SIZEOF (QUERY (sr <* shape_representation |
                NOT( SIZEOF(
       ['PART_204_BREP_PRODUCT_SCHEMA.ADVANCED_BREP_SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.ELEMENTARY_BREP_SHAPE_REPRESENTATION',
        'PART_204_BREP_PRODUCT_SCHEMA.FACETTED_BREP_SHAPE_REPRESENTATION']
           * TYPEOF (sr)) =1 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_geometry FOR (geometric_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (gri <* geometric_representation_item |
                 NOT (SIZEOF (USEDIN (gri,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_mapped_item FOR (mapped_item);
   WHERE
      WR1 : SIZEOF ( QUERY (mi <* mapped_item |
                 NOT (SIZEOF (USEDIN (mi,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_named_unit FOR (named_unit);
   WHERE
      WR1 : SIZEOF ( QUERY (nmu <* named |
                 NOT (SIZEOF (USEDIN (nmu,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_pre_defined_item FOR
                   (pre_defined_item);
   WHERE
      WR1 : SIZEOF ( QUERY (pdi <* presentation_style_assignment |
                 NOT (SIZEOF (USEDIN (pdi,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_presentation_style FOR
```

```
                (presentation_style_assignment);
   WHERE
      WR1 : SIZEOF ( QUERY (psa <* presentation_style_assignment |
                 NOT (SIZEOF (USEDIN (psa,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_styled_item FOR (styled_item);
   WHERE
      WR1 : SIZEOF ( QUERY (si <* styled_item |
                 NOT (SIZEOF (USEDIN (si,'')) > 0 ))) = 0;
END_RULE;

RULE dependent_instantiation_of_topology FOR (topological_representation_item);
   WHERE
      WR1 : SIZEOF ( QUERY (tri <* topological_representation_item |
                 NOT (SIZEOF (USEDIN (tri,'')) > 0 ))) = 0;
END_RULE;

RULE global_units_required FOR (global_unit_assigned_context);
   WHERE
      WR1 : SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                 NOT (SIZEOF (guac.units) <= 2))) = 0;
      WR2 : SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                 NOT (SIZEOF (QUERY( u <* guac.units |
                        'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT'
                               IN TYPEOF (u))) =1 ))) = 0;
      WR3:  SIZEOF ( QUERY (guac <* global_unit_assigned_context |
                 NOT (SIZEOF (QUERY( u <* guac.units |
                 NOT (SIZEOF (QUERY (other_u <* guac.units - u |
                 'PART_204_BREP_PRODUCT_SCHEMA.LENGTH_UNIT' IN TYPEOF(u))
                  AND
                   'PART_204_BREP_PRODUCT_SCHEMA.PLANE_ANGLE_UNIT]' IN
                       TYPEOF (other_u))))) = 0))) = 0;
END_RULE;

RULE no_complex_subtypes FOR(geometric_representation_item);
   WHERE
    WR1: SIZEOF (QUERY (gri <* geometric_representation_item |
    NOT (SIZEOF (TYPEOF(gri) * ['PART_204_BREP_PRODUCT_SCHEMA.ANNOTATION_TEXT',
               'PART_204_BREP_PRODUCT_SCHEMA.CURVE',
                'PART_204_BREP_PRODUCT_SCHEMA.DIRECTION',
           'PART_204_BREP_PRODUCT_SCHEMA.EDGE_CURVE',
           'PART_204_BREP_PRODUCT_SCHEMA.FACE_SURFACE',
            'PART_204_BREP_PRODUCT_SCHEMA.PLACEMENT',
             'PART_204_BREP_PRODUCT_SCHEMA.POINT',
           'PART_204_BREP_PRODUCT_SCHEMA.POLY_LOOP',
             'PART_204_BREP_PRODUCT_SCHEMA.SOLID_MODEL',
              'PART_204_BREP_PRODUCT_SCHEMA.SURFACE',
               'PART_204_BREP_PRODUCT_SCHEMA.VECTOR',
           'PART_204_BREP_PRODUCT_SCHEMA.VERTEX_POINT']) < 2 ))) = 0;
 END_RULE;
```

```
RULE product_context_mechanical FOR (product_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pc <* product_context |
                 NOT ('PART_204_BREP_PRODUCT_SCHEMA.MECHANICAL_CONTEXT'
            IN TYPEOF(pc)))) = 0;
END_RULE;

RULE product_definition_context_design FOR (product_definition_context);
   WHERE
      WR1 : SIZEOF ( QUERY (pdc <* product_definition_context |
                 NOT ('PART_204_BREP_PRODUCT_SCHEMA.DESIGN_CONTEXT'
                   IN TYPEOF(pc)))) = 0;
END_RULE;

 RULE three_dimensional_geometry FOR(geometric_representation_item);
   WHERE
    WR1: SIZEOF(QUERY(gri <* geometric_representation_item |
            (gri.dim = 2) AND NOT (SIZEOF (
            ['PART_204_BREP_PRODUCT_SCHEMA.PLANAR_BOX',
            'PART_204_BREP_PRODUCT_SCHEMA.ANNOTATION_TEXT'] *
              TYPEOF(gri)) =1 ))) = 0;
 END_RULE;


(* Functions from measures *)

FUNCTION dimensions_for_si_unit (n : si_unit_name) : dimensional_exponents;

  CASE n  OF
    metre          : RETURN (dimensional_exponents
                              (1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    gram           : RETURN (dimensional_exponents
                              (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    second         : RETURN (dimensional_exponents
                              (0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0));
    ampere         : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0));
    kelvin         : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    mole           : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0));
    candela        : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    radian         : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    steradian      : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0));
    hertz          : RETURN (dimensional_exponents
                              (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    newton         : RETURN (dimensional_exponents
                              (1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    pascal         : RETURN (dimensional_exponents
                              (-1.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
```

```
    joule          : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    watt           : RETURN (dimensional_exponents
                              (2.0, 1.0, -3.0, 0.0, 0.0, 0.0, 0.0));
    coulomb        : RETURN (dimensional_exponents
                              (0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0));
    volt           : RETURN (dimensional_exponents
                              (2.0, 1.0, -3.0, -1.0, 0.0, 0.0, 0.0));
    farad          : RETURN (dimensional_exponents
                              (-2.0, -1.0, 4.0, 1.0, 0.0, 0.0, 0.0));
    ohm            : RETURN (dimensional_exponents
                              (2.0, 1.0, -3.0, -2.0, 0.0, 0.0, 0.0));
    siemens        : RETURN (dimensional_exponents
                              (-2.0, -1.0, 3.0, 2.0, 0.0, 0.0, 0.0));
    weber          : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    tesla          : RETURN (dimensional_exponents
                              (0.0, 1.0, -2.0, -1.0, 0.0, 0.0, 0.0));
    henry          : RETURN (dimensional_exponents
                              (2.0, 1.0, -2.0, -2.0, 0.0, 0.0, 0.0));
    degree_Celsius : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0));
    lumen          : RETURN (dimensional_exponents
                              (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    lux            : RETURN (dimensional_exponents
                              (-2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0));
    becquerel      : RETURN (dimensional_exponents
                              (0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0));
    gray           : RETURN (dimensional_exponents
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
    sievert        : RETURN (dimensional_exponents
                              (2.0, 0.0, -2.0, 0.0, 0.0, 0.0, 0.0));
  END_CASE;
END_FUNCTION;


  FUNCTION msb_shells (brep: manifold_solid_brep;
                       schema_name : STRING) :
                       SET [1:?] OF  closed_shell;
     IF (schema_name + '.BREP_WITH_VOIDS' IN TYPEOF (brep)) THEN
       RETURN (brep\brep_with_voids.voids + brep.outer);
     ELSE
       RETURN([brep.outer]);
     END_IF;
  END_FUNCTION;

FUNCTION build_axes(axis, ref_direction : direction) :
                                        LIST [3:3] OF direction;
   LOCAL
     u : LIST[3:3] OF direction;
   END_LOCAL;

   u[3] := NVL(normalise(axis), direction([0.0,0.0,1.0]));
```

```
  u[1] := first_proj_axis(u[3],ref_direction);
  u[2] := normalise(cross_product(u[3],u[1]));
  RETURN(u);
END_FUNCTION;

 FUNCTION first_proj_axis(z_axis, arg : direction) : direction;
  LOCAL
    x_axis : direction;
    v      : direction;
  END_LOCAL;

  IF NOT EXISTS(z_axis) OR arg.dim <> 3 THEN
    x_axis := ?;
  ELSE
    z_axis := normalise(z_axis);

    IF NOT EXISTS(arg) THEN
      IF (z_axis <> direction([1.0,0.0,0.0])) THEN
        v := [1.0,0.0,0.0];
      ELSE
        v := [0.0,1.0,0.0];
      END_IF;
    ELSE
      IF ((cross_product(arg,z_axis).magnitude) = 0.0) THEN
        x_axis := ?;
      ELSE
        v := arg;
      END_IF;
    END_IF;

    x_axis := scalar_times_vector(dot_product(v, z_axis), z_axis);
    x_axis := vector_diff(v, x_axis);
    x_axis := normalise(x_axis);
  END_IF;
  RETURN(x_axis);
END_FUNCTION;


FUNCTION cross_product (arg1, arg2 : direction) : vector;
  LOCAL
    mag    : REAL;
    res    : direction;
    v1,v2  : LIST[3:3] OF REAL;
    result : vector;
  END_LOCAL;

  IF ( NOT EXISTS (arg1) OR (arg1.dim = 2)) OR
     ( NOT EXISTS (arg2) OR (arg2.dim = 2)) THEN
    RETURN(?);
  ELSE
    BEGIN
      v1  := normalise(arg1).direction_ratios;
      v2  := normalise(arg2).direction_ratios;
```

```
      res.direction_ratios[1] := (v1[2]*v2[3] - v1[3]*v2[2]);
      res.direction_ratios[2] := (v1[3]*v2[1] - v1[1]*v2[3]);
      res.direction_ratios[3] := (v1[1]*v2[2] - v1[2]*v2[1]);
      mag := 0.0;

      REPEAT i := 1 TO 3;
        mag := mag + res.direction_ratios[i]*res.direction_ratios[i];
      END_REPEAT;

      result.orientation := res;
      result.magnitude := SQRT(mag);
      RETURN(result);
    END;
  END_IF;
END_FUNCTION;

FUNCTION dot_product(arg1, arg2 : direction) : REAL;
  LOCAL
    scalar : REAL;
    vec1, vec2: direction;
    ndim : INTEGER;
  END_LOCAL;

  IF NOT EXISTS (arg1) OR NOT EXISTS (arg2) THEN
    scalar := ?;
    (* When function is called with invalid data a NULL result is returned *)
  ELSE
    IF (arg1.dim <> arg2.dim) THEN
      scalar := ?;
    (* When function is called with invalid data a NULL result is returned *)
    ELSE
      BEGIN
        vec1   := normalise(arg1);
        vec2   := normalise(arg2);
        ndim   := arg1.dim;
        scalar := 0.0;
        REPEAT  i := 1 TO ndim;
          scalar := scalar +
                       vec1.direction_ratios[i]*vec2.direction_ratios[i];
        END_REPEAT;
      END;
      RETURN (scalar);
    END_IF;
  END_IF;
END_FUNCTION;


FUNCTION normalise (arg : vector_or_direction) : vector_or_direction;
  LOCAL
    ndim   : INTEGER;
    v      : direction;
    result : vector_or_direction;
    vec    : vector;
```

```
    mag    : REAL;
  END_LOCAL;

  IF NOT EXISTS (arg) THEN
    result := ?;
    (* When function is called with invalid data a NULL result is returned *)
  ELSE
    ndim := arg.dim;
    IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF(arg) THEN
      BEGIN
        vec := arg;
        v := arg.orientation;

        IF arg.magnitude = 0.0 THEN
          RETURN(?);
        ELSE
          vec.magnitude := 1.0;
        END_IF;
      END;
    ELSE
      v := arg;
    END_IF;
    mag := 0.0;

    REPEAT  i := 1 TO ndim;
      mag := mag + v.direction_ratios[i]*v.direction_ratios[i];
    END_REPEAT;

    IF mag > 0.0 THEN
      mag := SQRT(mag);

      REPEAT  i := 1 TO ndim;
        v.direction_ratios[i] := v.direction_ratios[i]/mag;
      END_REPEAT;

      IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF(arg) THEN
        vec.orientation := v;
        result := vec;
      ELSE
        result := v;
      END_IF;
    ELSE
      RETURN(?);
    END_IF;
    RETURN (result);
  END_IF;
END_FUNCTION;


FUNCTION scalar_times_vector (scalar : NUMBER; vec : vector_or_direction)
                                     : vector;
  LOCAL
    v      : direction;
```

```
    mag    : REAL;
    result : vector;
  END_LOCAL;

  IF NOT EXISTS (scalar) OR NOT EXISTS (vec) THEN
    result := ?;
  (* When function is called with invalid data a NULL result is returned *)
  ELSE
    IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF (vec) THEN
      v   := vec.orientation;
      mag := scalar * vec.magnitude;
    ELSE
      v   := vec;
      mag := scalar;
    END_IF;

    result.orientation := normalise(v);
    result.magnitude   := mag;
    RETURN (result);
  END_IF;
END_FUNCTION;

FUNCTION vector_sum(arg1, arg2 : vector_or_direction) : vector;
  LOCAL
    result           : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim             : INTEGER;
  END_LOCAL;

  IF ((arg1.dim <> arg2.dim) OR (NOT EXISTS (arg1)) OR (NOT EXISTS (arg2)))
      THEN
    result := ?;
  (* When function is called with invalid data a NULL result is returned *)
  ELSE
    BEGIN
      IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1.0;
        vec1 := arg1;
      END_IF;

      IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF(arg2) THEN
        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1.0;
        vec2 := arg2;
      END_IF;
```

```
      vec1 := normalise (vec1);
      vec2 := normalise (vec2);
      ndim := SIZEOF(vec1.direction_ratios);

      REPEAT i := 1 TO ndim;
        res.direction_ratios[i] := mag1*vec1.direction_ratios[i] +
                                   mag2*vec2.direction_ratios[i];
        mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
      END_REPEAT;

      result.magnitude := SQRT(mag);
      result.orientation := res;
    END;
    RETURN (result);
  END_IF;
END_FUNCTION;


FUNCTION vector_diff(arg1, arg2 : vector_or_direction) : vector;
  LOCAL
    result          : vector;
    res, vec1, vec2 : direction;
    mag, mag1, mag2 : REAL;
    ndim            : INTEGER;
  END_LOCAL;

  IF ((arg1.dim <> arg2.dim) OR (NOT EXISTS (arg1)) OR (NOT EXISTS (arg2)))
      THEN
    result := ?;
  (* When function is called with invalid data a NULL result is returned *)
  ELSE
    BEGIN
      IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF(arg1) THEN
        mag1 := arg1.magnitude;
        vec1 := arg1.orientation;
      ELSE
        mag1 := 1.0;
        vec1 := arg1;
      END_IF;

      IF 'AIC_TOP_BND_SURF.VECTOR' IN TYPEOF(arg2) THEN
        mag2 := arg2.magnitude;
        vec2 := arg2.orientation;
      ELSE
        mag2 := 1.0;
        vec2 := arg2;
      END_IF;

      vec1 := normalise (vec1);
      vec2 := normalise (vec2);
      ndim := SIZEOF(vec1.direction_ratios);

      REPEAT i := 1 TO ndim;
```

```
        res.direction_ratios[i] := mag1*vec1.direction_ratios[i] -
                                        mag2*vec2.direction_ratios[i];
        mag := mag + (res.direction_ratios[i]*res.direction_ratios[i]);
      END_REPEAT;

      result.magnitude := SQRT(mag);
      result.orientation := res;
    END;
    RETURN (result);
  END_IF;
END_FUNCTION;


FUNCTION constraints_param_bspl(degree, up_knots, up_cp : INTEGER;
                                              knot_mult : LIST OF INTEGER;
                       knots : LIST OF parameter_value) : BOOLEAN;
  LOCAL
    result  : BOOLEAN := TRUE;
    k,l,sum : INTEGER;
  END_LOCAL;

  (* Find sum of knot multiplicities. *)
  sum := knot_mult[1];

  REPEAT i := 2 TO up_knots;
    sum := sum + knot_mult[i];
  END_REPEAT;

  (* Check limits holding for all B-spline parametrisations *)
  IF (degree < 1) OR (up_knots < 2) OR (up_cp < degree) OR
        (sum <> (degree + up_cp + 2)) THEN
    result := FALSE;
    RETURN(result);
  END_IF;

  k := knot_mult[1];

  IF (k < 1) OR (k > degree + 1) THEN
    result := FALSE;
    RETURN(result);
  END_IF;

  REPEAT i := 2 TO up_knots;
    IF (knot_mult[i] < 1) OR (knots[i] <= knots[i-1]) THEN
      result := FALSE;
      RETURN(result);
    END_IF;

    k := knot_mult[i];

    IF (i < up_knots) AND (k > degree) THEN
      result := FALSE;
      RETURN(result);
```

```
    END_IF;

    IF (i = up_knots) AND (k > degree + 1) THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
END_FUNCTION;

FUNCTION curve_weights_positive(b: rational_b_spline_curve) : BOOLEAN;
  LOCAL
    result : BOOLEAN := TRUE;
  END_LOCAL;

  REPEAT i := 0 TO b.upper_index_on_control_points;
    IF b.weights[i] <= 0.0  THEN
      result := FALSE;
      RETURN(result);
    END_IF;
  END_REPEAT;
  RETURN(result);
END_FUNCTION;

FUNCTION surface_weights_positive(b: rational_b_spline_surface) : BOOLEAN;
  LOCAL
    result        : BOOLEAN := TRUE;
  END_LOCAL;

  REPEAT i := 0 TO b.u_upper;
    REPEAT j := 0 TO b.v_upper;
      IF (b.weights[i][j] <= 0.0)  THEN
        result := FALSE;
        RETURN(result);
      END_IF;
    END_REPEAT;
  END_REPEAT;
  RETURN(result);
END_FUNCTION;



FUNCTION list_to_array(lis : LIST [0:?] OF GENERIC : T;
                     low,u : INTEGER) : ARRAY[low:u] OF GENERIC : T;
  LOCAL
    n   : INTEGER;
    res : ARRAY [low:u] OF GENERIC : T;
  END_LOCAL;

  n := SIZEOF(lis);
  IF (n <> (u-low +1)) THEN
    RETURN(?);
  ELSE
```

```
    REPEAT i := 1 TO n;
      res[low+i-1] := lis[i];
    END_REPEAT;
    RETURN(res);
  END_IF;
END_FUNCTION;

FUNCTION list_to_set(l : LIST [0:?] OF GENERIC:T) : SET OF GENERIC:T;
  LOCAL
    s : SET OF GENERIC:T := [];
  END_LOCAL;

  REPEAT i := 1 TO SIZEOF(l);
    s := s + l[i];
  END_REPEAT;

  RETURN(s);
END_FUNCTION;

FUNCTION make_array_of_array(lis : LIST[1:?] OF LIST [1:?] OF GENERIC : T;
                             low1, u1, low2, u2 : INTEGER):
                ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
  LOCAL
    n1,n2 : INTEGER;
    res   : ARRAY[low1:u1] OF ARRAY [low2:u2] OF GENERIC : T;
    resl  : LIST[1:?] OF ARRAY [low2:u2] OF GENERIC : T;
  END_LOCAL;

  (* Check input dimensions for consistency *)
  n1 := SIZEOF(lis);
  n2 := SIZEOF(lis[1]);

  IF (n1 <> (u1 -low1 + 1)) AND (n2 <> (u2 - low2 + 1)) THEN
    RETURN(?);
  END_IF;

  REPEAT i := 1 TO n1;
    IF (SIZEOF(lis[i]) <> n2) THEN
      RETURN(?);
    END_IF;
  END_REPEAT;

  (* Build a list of sub-arrays *)
  REPEAT i := 1 TO  n1;
    RESL[i] := list_to_array(lis[i],low2,u2);
  END_REPEAT;

  res := list_to_array(resl,low1,u1);
  RETURN(res);
END_FUNCTION;

FUNCTION boolean_choose (b : boolean;
```

```
           choice1, choice2 : generic) : generic;

   IF b THEN
     RETURN (choice1);
   ELSE
     RETURN (choice2);
   END_IF;
 END_FUNCTION;



  FUNCTION path_head_to_tail(a_path : path) : boolean;
   LOCAL
     n : INTEGER;
     p : BOOLEAN := TRUE;
   END_LOCAL;

     n := SIZEOF (a_path.edge_list);
     REPEAT i := 2 TO n;
       p := p AND (a_path.edge_list[i-1].edge_end :=:
                   a_path.edge_list[i].edge_start);
     END_REPEAT;

     RETURN (p);
 END_FUNCTION;

 FUNCTION list_face_loops(f: face) : LIST[0:?] OF loop;
   LOCAL
     loops : LIST[0:?] OF loop := [];
   END_LOCAL;

   REPEAT i := 1 TO SIZEOF(f.bounds);
     loops := loops +(f.bounds[i].bound);
   END_REPEAT;

   RETURN(loops);
 END_FUNCTION;

FUNCTION list_loop_edges(l: loop): LIST[0:?] OF edge;
   LOCAL
     edges : LIST[0:?] OF edge := [];
   END_LOCAL;

   IF 'AIC_TOP_BND_SURF.EDGE_LOOP' IN TYPEOF(l) THEN
     REPEAT i := 1 TO SIZEOF(l\path.edge_list);
       edges := edges + [l\path.edge_list[i].edge_element];
     END_REPEAT;
   END_IF;

   RETURN(edges);
 END_FUNCTION;

  FUNCTION mixed_loop_type_set(l: SET[0:?] OF loop): BOOLEAN;
```

```
   LOCAL
     i             : INTEGER;
     poly_loop_type: BOOLEAN;
   END_LOCAL;
   IF(SIZEOF(l) <= 1) THEN
     RETURN(FALSE);
   END_IF;
   poly_loop_type := ('AIC_TOP_BND_SURF.POLY_LOOP' IN TYPEOF(l[1]));
   REPEAT i := 2 TO SIZEOF(l);
     IF(('AIC_TOP_BND_SURF.POLY_LOOP' IN TYPEOF(l[i])) :<>: poly_loop_type) THEN
        RETURN(TRUE);
      END_IF;
   END_REPEAT;
   RETURN(FALSE);
 END_FUNCTION;

FUNCTION conditional_reverse (p       : BOOLEAN;
                              an_item : reversible_topology)
                                      : reversible_topology;
  IF p THEN
    RETURN (an_item);
  ELSE
    RETURN (topology_reversed (an_item));
  END_IF;
END_FUNCTION;


FUNCTION topology_reversed (an_item : reversible_topology)
                                    : reversible_topology;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.EDGE' IN TYPEOF (an_item)) THEN
    RETURN (edge_reversed (an_item));
  END_IF;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.PATH' IN TYPEOF (an_item)) THEN
    RETURN (path_reversed (an_item));
  END_IF;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.FACE_BOUND' IN TYPEOF (an_item)) THEN
    RETURN (face_bound_reversed (an_item));
  END_IF;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.FACE' IN TYPEOF (an_item)) THEN
    RETURN (face_reversed (an_item));
  END_IF;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.SHELL' IN TYPEOF (an_item)) THEN
    RETURN (shell_reversed (an_item));
  END_IF;

  IF ('SET' IN TYPEOF (an_item)) THEN
    RETURN (set_of_topology_reversed (an_item));
  END_IF;
```

```
   IF ('LIST' IN TYPEOF (an_item)) THEN
     RETURN (list_of_topology_reversed (an_item));
   END_IF;

   RETURN (?);
 END_FUNCTION;
FUNCTION shell_reversed (a_shell : shell) : shell;
   LOCAL
     the_reverse : shell;
   END_LOCAL;

   IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_OPEN_SHELL' IN
                                        TYPEOF (a_shell) ) THEN
     the_reverse := oriented_open_shell(a_shell.open_shell_element,
                               (NOT (a_shell.orientation)));
   ELSE
     IF ('PART_204_BREP_PRODUCT_SCHEMA.OPEN_SHELL' IN TYPEOF (a_shell) ) THEN
       the_reverse := oriented_open_shell (a_shell, FALSE);
     ELSE
       IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_CLOSED_SHELL' IN
                                             TYPEOF (a_shell) ) THEN
         the_reverse := oriented_closed_shell
                              (a_shell.closed_shell_element,FALSE);
       ELSE
         IF ('PART_204_BREP_PRODUCT_SCHEMA.CLOSED_SHELL' IN
                                             TYPEOF (a_shell) ) THEN
           the_reverse := oriented_closed_shell (a_shell, FALSE);
         ELSE
           the_reverse := ?;
         END_IF;
       END_IF;
     END_IF;
  END_IF;

  RETURN (the_reverse);
 END_FUNCTION;

 FUNCTION face_reversed (a_face : face) : face;
   LOCAL
     the_reverse : face;
   END_LOCAL;

   IF ('PART_204_BREP_PRODUCT_SCHEMA.ORIENTED_FACE' IN
                                      TYPEOF (a_face) ) THEN
     the_reverse := oriented_face(a_face.face_element,
                                  (NOT (a_face.orientation)));
   ELSE
     the_reverse := oriented_face (a_face, FALSE);
   END_IF;

   RETURN (the_reverse);
```

```
END_FUNCTION;


FUNCTION face_bound_reversed (a_face_bound : face_bound) : face_bound;
  LOCAL
    the_reverse : face_bound;
  END_LOCAL;

  IF ('PART_204_BREP_PRODUCT_SCHEMA.FACE_OUTER_BOUND' IN
                                  TYPEOF (a_face_bound) ) THEN
    the_reverse := face_outer_bound(a_face_bound.bound,
                                     (NOT (a_face_bound.orientation)));
  ELSE
    the_reverse := face_bound(a_face_bound.bound,
                              (NOT (a_face_bound.orientation)));
  END_IF;

  RETURN (the_reverse);
END_FUNCTION;

FUNCTION set_of_topology_reversed (a_set : set_of_reversible_topology_item)
                                        : set_of_reversible_topology_item;
  LOCAL
    the_reverse : set_of_reversible_topology_item;
  END_LOCAL;

  the_reverse := [];
  REPEAT i := 1 TO SIZEOF (a_set);
    the_reverse := the_reverse + topology_reversed (a_set [i]);
  END_REPEAT;

  RETURN (the_reverse);
END_FUNCTION;

FUNCTION dimension_of(item : geometric_representation_item) :
 dimension_count;
 LOCAL
   x : SET OF representation;
   y : representation_context;
   z : SET OF definitional_representation_item;
 END_LOCAL;
 x := using_representations(item);

 IF SIZEOF(x) > 0 THEN
   y := x[1].context_of_items;
   RETURN (y\geometric_representation_context.coordinate_space_dimension);

 ELSE
   z := using_definitional_representation_items(item);
   y := z[1].context_of_items;
   RETURN (y\geometric_representation_context.coordinate_space_dimension);
 END_IF;
```

```
END_FUNCTION;


FUNCTION using_representations (item : representation_item)
  : SET OF representation;
  LOCAL
    results            : SET OF representation;
    intermediate_items : SET OF representation_item;
    i                  : INTEGER;
  END_LOCAL;
  results := USEDIN(item,'AIC_TOP_BND_SURF.REPRESENTATION.ITEMS');
  intermediate_items := QUERY(z <* USEDIN(item , '') |
    'AIC_TOP_BND_SURF.REPRESENTATION_ITEM' IN TYPEOF(z));
  IF SIZEOF(intermediate_items) > 0 THEN
    REPEAT i := 1 TO HIINDEX(intermediate_items);
      results := results + using_representations(intermediate_items[i]);
    END_REPEAT;
  END_IF;
  RETURN (results);
END_FUNCTION;

FUNCTION using_definitional_representation_items(item : representation_item)
  : SET OF definitional_representation_item;
  LOCAL
    results            : SET OF definitional_representation_item;
    intermediate_items : SET OF representation_item;
    i                  : INTEGER;
  END_LOCAL;

  results := USEDIN(item,
             'AIC_TOP_BND_SURF.DEFINITIONAL_REPRESENTATION_ITEM.ITEMS');
  intermediate_items := QUERY(z <* USEDIN(item , '') |
    'AIC_TOP_BND_SURF.REPRESENTATION_ITEM' IN TYPEOF(z));
  IF SIZEOF(intermediate_items) > 0 THEN
    REPEAT i := 1 TO HIINDEX(intermediate_items);
      results := results + using_definitional_representation_items
        (intermediate_items[i]);
    END_REPEAT;
  END_IF;
  RETURN (results);
END_FUNCTION;


FUNCTION acyclic_mapped_representation
  (parent_set   : SET OF representation;
   children_set : SET OF representation_item) : BOOLEAN;
  LOCAL
    x,y : SET OF representation_item;
    i,j : INTEGER;
  END_LOCAL;

  -- Determine the subset of children_set that are mapped_items.
  x := QUERY(z <* children_set | 'REPRESENTATION_SCHEMA.MAPPED_ITEM'
```

```
      IN TYPEOF(z));

  -- Determine that the subset has elements.
  IF SIZEOF(x) > 0 THEN

    -- Check each element of the set.
    REPEAT i := 1 TO HIINDEX(x);


      -- If the selected element maps a representation in the parent_set,
      -- return false.
      IF x[i]\mapped_item.mapping_source.mapped_representation
        IN parent_set THEN
        RETURN (FALSE);
      END_IF;

      -- Recursively check the items of the mapped_rep.
      If NOT acyclic_mapped_representation
        (parent_set + x[i]\mapped_item.mapping_source.mapped_representation,
        x[i]\mapped_item.mapping_source.mapped_representation.items) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;

  -- Determine the subset of children_set that are not mapped_items.
  x := children_set - x;

  -- Determine that the subset has elements.
  IF SIZEOF(x) > 0 THEN

    -- For each element of the set:
    REPEAT i := 1 TO HIINDEX(x);

      -- Determine the set of representation_items it references.
      y := QUERY(z <* USEDIN(x[i], '') |
          'REPRESENTATION_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

      -- Recursively check these in case they might be an offending
      -- mapped_item.  Return false for any errors encountered.
      If NOT acyclic_mapped_representation(parent_set, y) THEN
        RETURN (FALSE);
      END_IF;
    END_REPEAT;
  END_IF;

  -- Return true when all elements are checked and
  -- no error conditions found.
  RETURN (TRUE);
END_FUNCTION;
```

```
FUNCTION acyclic_product_definition_relationship
  (relation          : product_definition_relationship;
   relatives          : SET OF product_definition;
   specific_relation : STRING) : LOGICAL;

  LOCAL
    x              : SET OF product_definition_relationship;
    i              : INTEGER;
    local_relatives : SET OF product_definition;
  END_LOCAL;

  REPEAT i := 1 TO HIINDEX(relatives);
    IF relation.relating_product_definition :=: relatives[i] THEN
      RETURN(FALSE);
    END_IF;
  END_REPEAT;
  x := USEDIN (relation.relating_product_definition, specific_relation);
       local_relatives := relatives + relation.relating_product_definition;
  IF SIZEOF(x) > 0 THEN
    REPEAT i := 1 TO HIINDEX(x);
      IF NOT acyclic_product_definition_relationship
             (x[i], local_relatives, specific_relation) THEN
        RETURN(FALSE);
      END_IF;
    END_REPEAT;
  END_IF;
  RETURN(TRUE);
END_FUNCTION;


FUNCTION item_in_context
  (item  : representation_item;
   cntxt : representation_context) : BOOLEAN;
  LOCAL
    i : INTEGER;
    y : SET OF representation_item;
  END_LOCAL;

  -- If there is one or more representation using both the item and cntxt
  -- return true.
  IF SIZEOF(USEDIN(item,'REPRESENTATION_SCHEMA.REPRESENTATION.ITEMS')
    * cntxt.contexted_representations) > 0 THEN
    RETURN (TRUE);

    -- Else if there is one or more definitional_representation_items using
    -- both the item and the cntxt, return true.
  ELSE
    IF  SIZEOF(USEDIN(item,
      'GEOMETRY_SCHEMA.DEFINITIONAL_REPRESENTATION_ITEM.ITEMS')
      * cntxt.contexted_definitional_representation_items) > 0 THEN
      RETURN(TRUE);

    -- Else determine the set of representation_items that reference item.
```

```
    ELSE
      y := QUERY(z <* USEDIN (item , '') |
           'REPRESENTATION_SCHEMA.REPRESENTATION_ITEM' IN TYPEOF(z));

      -- Ensure that the set is not empty.
        IF SIZEOF(y) > 0 THEN

        -- For each element in the set
        REPEAT i := 1 TO HIINDEX(y);

          --  Recursively check to see it is an item in the input cntxt.
          IF item_in_context(y[i], cntxt) THEN
            RETURN (TRUE);
          END_IF;
        END_REPEAT;
      END_IF;
    END_IF;
  END_IF;

  -- Return false when all possible branches have been checked
  -- with no success.
  RETURN (FALSE);
END_FUNCTION;

FUNCTION mdp_get_text_literal_mapped_item_in_annotation_text
           (at : annotation_text, schema_name : STRING) :
           BAG [0:?] OF text_literal_mapped_item;

  LOCAL
    txlits: BAG OF text_literal_mapped_item := [];
    tsr   : text_string_representation :=
            at\mapped_item.mapping_source.mapped_representation;
  END_LOCAL;

  -- check whether the input has the right type
  IF NOT schema_name + '.ANNOTATION_TEXT' IN TYPEOF (at) THEN RETURN(txlits);
  END_IF;

  -- do for each member in the strings of the text_string_representation
  REPEAT i := 1 TO SIZEOF (tsr.strings);

    -- if the member is annotation_text, go into recursion
    IF schema_name + '.ANNOTATION_TEXT' IN TYPEOF (tsr.strings[i]) THEN
      txlits := txlits + mdp_get_text_literal_mapped_item_in_annotation_text
      (tsr.strings[i], schema_name);
    ELSE

    -- if the member is text_literal_mapped_item, collect it and go to next
    -- member
      IF schema_name + '.TEXT_LITERAL_MAPPED_ITEM' IN TYPEOF
      (tsr.strings[i]) THEN
        txlits := txlits + 1;
```

```
      ELSE

        -- if anything is wrong, return an empty bag
        txlits := [];
        RETURN(txlits);
      END_IF;
    END_IF;
  END_REPEAT;

  RETURN(txlits);

END_FUNCTION;

 END_SCHEMA; -- end PART_204_BREP_PRODUCT_SCHEMA

 (*
```

Annex B
(normative)

Entity and type abbreviations

Table B.1 provides the short names of entities specified in the AIM of this part of ISO 10303. Requirements on the use of the short names are found in the implementation methods included in ISO 10303.

Table B.1 – Short names of entities

Entity name	Short name
ADVANCED_BREP_SHAPE_REPRESENTATION	ABSR
ADVANCED_FACE	ADVFAC
ANNOTATION_OCCURRENCE	ANNOCC
ANNOTATION_TEXT	ANNTXT
ANNOTATION_TEXT_MAP	ANTXMP
ANNOTATION_TEXT_OCCURRENCE	ANTXOC
APPLICATION_CONTEXT	APPCNT
APPLICATION_CONTEXT_ELEMENT	APCNEL
ASSEMBLY_COMPONENT_USAGE	ASCMUS
AXIS1_PLACEMENT	AX1PLC
AXIS2_PLACEMENT_3D	A2PL3D
BACKGROUND_COLOUR	BCKCLR
BEZIER_CURVE	BZRCRV
BEZIER_SURFACE	BZRSRF
BOUNDED_CURVE	BNDCR
BOUNDED_SURFACE	BNDSRF
BREP_WITH_VOIDS	BRWTVD
B_SPLINE_CURVE	BSPCR
B_SPLINE_CURVE_WITH_KNOTS	BSCWK
B_SPLINE_SURFACE	BSPSR
B_SPLINE_SURFACE_WITH_KNOTS	BSSWK
CAMERA_IMAGE	CMRIMG
CAMERA_MODEL	CMRMDL
CAMERA_MODEL_D3	CMMDD3
CAMERA_USAGE	CMRUSG

Table B.1 (continued)

Entity name	Short name
CARTESIAN_POINT	CRTPNT
CARTESIAN_TRANSFORMATION_OPERATOR	CRTROP
CARTESIAN_TRANSFORMATION_OPERATOR_3D	CTO3
CIRCLE	CIRCLE
CLOSED_SHELL	CLSSHL
COLOUR	COLOUR
COLOUR_RGB	CLRRGB
COLOUR_SPECIFICATION	CLRSPC
CONIC	CONIC
CONICAL_SURFACE	CNCSRF
CONNECTED_FACE_SET	CNFCST
CONVERSION_BASED_UNIT	CNBSUN
CURVE	CURVE
CURVE_STYLE	CRVSTY
CURVE_STYLE_FONT	CRSTFN
CURVE_STYLE_FONT_PATTERN	CSFP
CURVE_STYLE_RENDERING	CRSTRN
CYLINDRICAL_SURFACE	CYLSRF
DESIGN_CONTEXT	DSGCNT
DIMENSIONAL_EXPONENTS	DMNEXP
DIRECTION	DRCTN
EDGE	EDGE
EDGE_CURVE	EDGCRV
EDGE_LOOP	EDGLP
ELEMENTARY_BREP_SHAPE_REPRESENTATION	EBSR
ELEMENTARY_SURFACE	ELMSRF
ELLIPSE	ELLPS
FACE	FACE
FACETTED_BREP	FCTBRP
FACETTED_BREP_SHAPE_REPRESENTATION	FBSR
FACE_BOUND	FCBND
FACE_OUTER_BOUND	FCOTBN
FACE_SURFACE	FCSRF
FUNCTIONALLY_DEFINED_TRANSFORMATION	FNDFTR

Table B.1 (continued)

Entity name	Short name
GEOMETRIC_REPRESENTATION_CONTEXT	GMRPCN
GEOMETRIC_REPRESENTATION_ITEM	GMRPIT
GLOBAL_UNIT_ASSIGNED_CONTEXT	GUAC
HYPERBOLA	HYPRBL
LENGTH_MEASURE_WITH_UNIT	LMWU
LENGTH_UNIT	LNGUNT
LINE	LINE
LOOP	LOOP
MANIFOLD_SOLID_BREP	MNSLBR
MAPPED_ITEM	MPPITM
MEASURE_WITH_UNIT	MSWTUN
MECHANICAL_CONTEXT	MCHCNT
MECHANICAL_DESIGN_NAME_ASSIGNMENT	MDNA
MECHANICAL_DESIGN_PRESENTATION_AREA	MDPA
MECHANICAL_DESIGN_PRESENTATION_REPRESENTATION	MDPR
MECHANICAL_DESIGN_PRE_DEFINED_TEXT_FONT	MDPDTF
MECHANICAL_DESIGN_PRE_DEFINED_TEXT_STYLE	MDPDTS
NAMED_UNIT	NMDUNT
NAME_ASSIGNMENT	NMASS
ORIENTED_CLOSED_SHELL	ORCLSH
ORIENTED_EDGE	ORNEDG
ORIENTED_FACE	ORNFC
PARABOLA	PRBL
PATH	PATH
PLACEMENT	PLCMNT
PLANAR_BOX	PLNBX
PLANAR_EXTENT	PLNEXT
PLANE	PLANE
PLANE_ANGLE_MEASURE_WITH_UNIT	PAMWU
PLANE_ANGLE_UNIT	PLANUN
POINT	POINT
POINT_STYLE	PNTSTY
POLYLINE	PLYLN
POLY_LOOP	PLYLP

Table B.1 (continued)

Entity name	Short name
PRESENTATION_AREA	PRSAR
PRESENTATION_LAYER_ASSIGNMENT	PRLYAS
PRESENTATION_LAYER_USAGE	PRLYUS
PRESENTATION_REPRESENTATION	PRSRPR
PRESENTATION_SIZE	PRSSZ
PRESENTATION_STYLE_ASSIGNMENT	PRSTAS
PRESENTATION_STYLE_BY_CONTEXT	PSBC
PRESENTATION_VIEW	PRSVW
PRE_DEFINED_ITEM	PRDFIT
PRE_DEFINED_PRESENTATION_STYLE	PDPS
PRE_DEFINED_TEXT_FONT	PDTF
PRODUCT	PRDCT
PRODUCT_CATEGORY	PRDCTG
PRODUCT_CONTEXT	PRDCNT
PRODUCT_DATA_REPRESENTATION_VIEW	PDRV
PRODUCT_DATA_REPRESENTATION_VIEW_WITH_HLHSR	PDRVWH
PRODUCT_DEFINITION	PRDDFN
PRODUCT_DEFINITION_CONTEXT	PRDFCN
PRODUCT_DEFINITION_RELATIONSHIP	PRDFRL
PRODUCT_DEFINITION_SHAPE	PRDFSH
PRODUCT_DEFINITION_USAGE	PRDFUS
PRODUCT_RELATED_PRODUCT_CATEGORY	PRPC
PRODUCT_VERSION	PRDVRS
QUASI_UNIFORM_CURVE	QSUNCR
QUASI_UNIFORM_SURFACE	QSUNSR
RATIONAL_B_SPLINE_CURVE	RBSC
RATIONAL_B_SPLINE_SURFACE	RBSS
REPRESENTATION	RPRSNT
REPRESENTATION_CONTEXT	RPRCNT
REPRESENTATION_DEPENDENT_STYLED_ITEM	RDSI
REPRESENTATION_ITEM	RPRITM
REPRESENTATION_ITEM_WITHOUT_STYLE	RIWS
REPRESENTATION_MAP	RPRMP
REPRESENTATION_RELATIONSHIP	RPRRLT
REPRESENTATION_RELATIONSHIP_WITH_TRANSFORMATION	RPRRLT
SHAPE_DEFINITION_REPRESENTATION	SHDFRP
SHAPE_REPRESENTATION	SHPRPR
SI_UNIT	SUNT
SOLID_MODEL	SLDMDL
SPHERICAL_SURFACE	SPHSRF
STYLED_ITEM	STYITM

Table B.1 (concluded)

Entity name	Short name
SURFACE	SRFC
SURFACE_OF_LINEAR_EXTRUSION	SL
SURFACE_OF_REVOLUTION	SROFRV
SURFACE_RENDERING_PROPERTIES	SRRNPR
SURFACE_SIDE_STYLE	SRSDST
SURFACE_STYLE_BOUNDARY	SRSTBN
SURFACE_STYLE_CONTROL_GRID	SSCG
SURFACE_STYLE_PARAMETER_LINE	SSPL
SURFACE_STYLE_RENDERING	SRSTRN
SURFACE_STYLE_SEGMENTATION_CURVE	SSSC
SURFACE_STYLE_SILHOUETTE	SRSTSL
SURFACE_STYLE_USAGE	SRSTUS
SWEPT_SURFACE	SWPSRF
SYMBOL_REPRESENTATION	SYMRPR
TEXT_LITERAL_MAPPED_ITEM	TLMI
TEXT_LITERAL_SYMBOL	TXLTSY
TEXT_STRING_REPRESENTATION	TXSTRP
TEXT_SYMBOL	TXTSYM
TOPOLOGICAL_REPRESENTATION_ITEM	TPRPIT
TOROIDAL_SURFACE	TRDSRF
UNIFORM_CURVE	UNFCRV
UNIFORM_SURFACE	UNFSRF
VECTOR	VECTOR
VERTEX	VERTEX
VERTEX_LOOP	VRTLP
VERTEX_POINT	VRTPNT
VIEW_DEPENDENT_ANNOTATION_REPRESENTATION	VDAR
VIEW_VOLUME	VWVLM

NOTE – This list of short names is provisional and may be amended when a computer generated list is available.

Annex C
(normative)

PICS (Protocol Implementation Conformance Statement) proforma

This clause lists the optional elements of this part of ISO 10303. An implementation may choose to support any combination of these optional elements. However, certain combinations of options are likely to be implemented together. These combinations are called conformance classes and are described in the subclauses of this annex.

This annex is in the form of a questionnaire. This questionnaire is intended to be filled out by the implementor and may be used in preperation for conformance testing by a testing laboratory. The completed PICS proforma is referred to as a PICS.

NOTE 1 – This annex has to be completed.

Annex D
(normative)

Implementation method specific requirements

The primary implementation form supported by this AP is an exchange structure implementation using the exchange structure specified in ISO 10303-21.

The header section of an exchange structure conforming to ISO 10303-21 shall identify the AIM schema and the conformance class of the implementation to identify the types of B-rep model represented in the file.

NOTE – A simple example of such an exchange structure is given in Annex K.

Other forms of implementation, such as the Standard Data Access Interface (SDAI) conforming to ISO 10303-22 may be supported but no specific requirements are given for these.

Annex E
(informative)

Application Activity Model (AAM)

The application activity model (AAM) is provided to aid in understanding the scope and information requirements defined in this application protocol. The model is presented as a set of definitions of the activities and the data and a set of activity figures. The AAM covers activities which go beyond the scope of this application protocol.

The viewpoint of the application activity model is the use and exchange of boundary representation models in the mechanical engineering environment.

E.1 AAM Definitions

The following terms are used in the application activity model. Terms marked with an asterisk are outside the scope of this application protocol.

The definitions given in this annex do not supercede the definitions given in the main body of the text.

E.1.1 CAD model:

The computer internal description of product shape produced by a computer aided design system.

E.1.2 Calculation:*

The process of performing a mathematical computation.

E.1.3 Calculation system*:

A system which produces a mathematical model of some aspect of the physical structure or behaviour of a product.

E.1.4 Concept development*:

The process of producing the conceptual design.

E.1.5 Conceptual change request*:

A request for a change in the conceptual design of a product.

E.1.6 Conceptual design*:

The preliminary definition of the product with respect to technical, economic, ecological and strategic requirements.

E.1.7 Definitional change request*:

A request for a change in the definition of a product.

E.1.8 Design

The development of the shape of a product taking account of its functionality and physical properties.

E.1.9 Design change request:

A formal request to change any form, fit or function aspect of an existing design.

E.1.10 Development tool:*

A computer tool which assists in the development of a product definition.

E.1.11 Evaluation*:

The testing and validation of a product design, or the prototype of a product, against its requirements.

E.1.12 FEA system *

A computer system for the analysis of CAD models with respect to such properties as stress, dynamic behaviour, or heat transfer using the finite element method.

E.1.13 Holography*:

A process for producing three dimensional images using lasers.

E.1.14 Knowhow*:

The knowledge and experience of the people involved in the development process.

E.1.15 Machine tool*:

A mechanical device, usually for cutting metal, used in the manufacturing process.

E.1.16 Machine control code*:

A computer sensible code controlling a machine tool.

E.1.17 Management*:

The process of controlling the use of manpower and resources.

E.1.18 Manpower*:

The people, or human resources, involved in the process.

E.1.19 Manufacturing*:

The realisation of the actual product.

E.1.20 Marketing strategy*:

The planned method for selling the product.

E.1.21 Material*:

The matter from which the product is manufactured.

E.1.22 Measured data*:

Data which is obtained by measuring a product or physical model.

E.1.23 NC-machining*:

A manufacturing process of material removal using a numerically controlled machine tool.

E.1.24 Part design:

The process of defining the form, fit and function of a part.

E.1.25 Physical model*:

A representation of the product or part made of some material substance.

E.1.26 Planning tools*:

Methods and equipment for the support of planning tasks, e.g. process planning system.

E.1.27 Prototyping*:

The process of producing a first example of a product for evaluation.

E.1.28 Product plan*:

All available information about the manufacturing process for a product, including the sequence of manufacturing operations.

E.1.29 Product requirement specification*:

The collection of all known requirements and constraints which specify the functionality and characteristics of a product.

E.1.30 Product specification:

The description of the characteristics of a product.

E.1.31 Production planning*:

The process of producing the product plan.

E.1.32 Quality assurance*:

The process of ensuring that the product meets its specification.

E.1.33 Shape design:

The process of developing the precise specification of the geometric form of a product.

E.1.34 Sketch*:

An approximate, possibly hand-drawn, presentation of the form of a product.

E.1.35 Simulation*:

The imitation of a process, using a physical model, or using a computer model.

E.1.36 Standards*:

Formal documents, which may be international, national or company specific, which influence the design process to ensure commonality between products.

E.1.37 Stereolithography*:

A method of producing three dimensional physical models of a product.

E.1.38 Test plan*:

A plan for testing a product design, or prototype, to ensure that all requirements are met.

E.1.39 Visualisation:

The pictorial presentation of a product.

E.1.40 Visualisation system*:

A system which processes the product description data into a pictorial representation.

E.2 AAM Description

Mechanical engineering design and manufacturing use many different there are heterogeneous CAD and CAE and CIM systems. Different CAD models are inherently in these systems. The data exchange between those systems requires a neutral Standard rather than system dependent models or methods. In the following subclauses the mechanical design requirements are described.

E.2.1 Mechanical Design Requirements for Model Exchange

The different groups involved in the design, engineering, or manufacturing processes need to have access to the design data. The basic geometric form of a product is fixed during the conceptual design of the product. The details of the product shape may be added by a design group using a CAD system. The design represented in a CAD model may be transferred from system to system by means of a neutral or standard file. The transfer medium might be an electronic network (i.e. LAN, WAN), physical media, or some form of shared database.

The CAD model data is transferred to other groups within the company or transferred to another company for analysis or manufacturing purposes. The following scenarios should be considered for model exchange.

E.2.1.1 Design to Product Analysis and Simulation

Analysis and simulation of the product are used at different stages of the development process for early testing and validation of the design. Product data is received from the design groups via data transfer of CAD models and manufacturing data (e.g., NC programmes). The results of the prototype evaluation process (e.g., test protocols, requirements) are returned to the design department and may lead to a new iteration. The types of analysis performed during the validation process are dependent upon the functionality of the product and may include stress or thermal analysis or aerodynamic analysis on a given mechanical design.

E.2.1.2 Design to Design groups

A complex product may be designed with parts from different design and manufacturing sources. The product may be an assembly of designs from different companies. In this case the designs of individual parts may be exchanged between the design companies. A contractor may transfer to a subcontractor a 'hull of a design' which indicates the outer boundaries of parts in detail at only specific interfacing points but not the complete shape of the parts (e.g., a dimensioned box for a motor block with some details for the bearings). This scenario requires the transfer of the exact shape model.

E.2.1.3 Design to Prototype Manufacturing

Prototypes of the product are used at different stages of the development process for testing and validation of the design. The necessary data is received from the design departments via the data transfer of CAD models and manufacturing data (NC programmes etc.). The results of the prototype evaluation process (test protocols, requirements etc.) are returned to the design department and may lead to a new iteration of the design and validation process.

E.2.1.4 Design to Manufacturing Planning and Manufacturing

The manufacturing planning department does the detailed manufacturing process plan in order to manufacture a product. If the part is to be manufactured with metal cutting machinery then the appropriate technology (milling, drilling, turning and grinding) is selected and Numerical Controlled (NC) machine tool programs are set up. The data are derived from the CAD model (required data: geometry, topology, material, administrative data). The NC tool paths are generated in the NC programming system; collision checks with the machine tools and tool path simulations are performed. The result is a NC machine program which can be 'downloaded' to the NC machine. A similar procedure is performed if robots and robot programming systems are applied (e.g. for material supply to NC machines).

E.2.1.5 Design to Assembly Planning

In cases where robots are applied to assemble the products then a robot simulation program might be applied in order to visualize the assembly process prior to the physical assembly process. Assembly path simulations and collision checks are performed based on geometric assembly paths which are typically generated interactively in a simulation workstation. Final assembly fitting simulation is performed on given shapes of parts.

E.2.1.6 Design to Quality Assurance

Quality assurance today is performed in parallel to all design to analysis and manufacturing processes. The design data are used as reference data for any quality assurance in subsequent processes after the design.

EXAMPLES

9 – The design model is the basis for analysis (e.g., stress) by means of Finite Element models. The analysis results give data for various means of assuring quality (e.g., modification of the shape of a design).

10 – The design model is a reference for manufacturing inspection (e.g. when applying a coordinate measurement machine after a mechanical part is machined by a milling machine). If there is a deviation beyond a certain tolerance limit then the manufactured part is not acceptable.

E.2.1.7 Design to documentation

The design master model, resident in a computer and based on topology, geometry and other associated information is used to derive CAD drawing documentation. The 2D CAD documentation typically consists of orthogonal projections of the 3D model to 2D space: the results are 2D CAD drawings. These contain the geometry of the designed parts, explanatory text, dimensioning, hatching for sectioned areas, and administrative information. The sum of the design data is typically kept in a data management system.

E.2.1.8 Feedback to the design department

The initial design of a part which will be manufactured is typically not right the first time. Therefore model data typically have to be transferred back and modified in the design department after they have been transferred either to the manufacturing process planning or to the part analysis engineering departments. These departments give feedback to the design office in the form of change requests relating to the original CAD model data file. The process described above can of course be handled via a data base system which could be accessible from different departments. Other intermediate engineering or pre-engineering organisational entities might be involved in the design process and in its feedback loop.

E.3 Mechanical Design Requirements for model contents and completeness

The Mechanical Design area has a very large scope which is intentionally limited here to essential information blocks. Figure E.1 gives an overview on the conceptual structure of a Mechanical Design Product.

A mechanical design product model typically is composed of the following data:

E.3.1 Shape description (Geometry and topology)

The shape description of part models can have different levels of completeness and constraints and therefore different model descriptions are applied today:

- A: Volume based design (B-rep or CSG);
- B: Surface based design;
- C: Wireframe based design (curves).

The different geometric descriptions typically do not all exist simultaneously for one product rather they are set up during the design process in one of the above mentioned forms.

E.3.2 Draughting Data

Another kind of representation of a product is the classical technology of a drawing on paper. This is still the most commonly used methodology for design, manufacturing and archiving of product data. The drawings can be produced in a number of different ways:

- by using a pen and paper
- by using a pure 2D CAD draughting system
- from a 3D CAD model projected into 2D drawing space.

E.3.3 Physical property description

There are a variety of physical properties which could be attached to a part model, essential amongst these are the material properties. Material properties may be determined by a designer quite late in the phase of Detail Design. Each model part in this Application Protocol assumes the use of one homogeneous material. There might be other physical properties (e.g. surface roughness) assigned to parts but these are beyond the scope of this Application Protocol. The material property is included as representative of other properties which could be included later.

E.3.4 Part administrative data

This contains an aggregation of administrative data for a specific part. The administrative description has a reference to the shape description and might also have a reference to the Material description.

E.3.5 Assemblies

Assemblies are composed of either sub-assemblies and parts, or, of at least two parts. An assembly is typically represented as a tree of parts and sub-assemblies. The relationship between assemblies and sub-assemblies or parts contains positional and orientational information in three dimensional space.

E.3.6 Product Information Description

This section covers basic administrative data for managing the product through the design process and for the use of other departments in a manufacturing engineering company.

EXAMPLES

11 – identification of the product (name);

12 – ownership (company, department, designer,manager);

13 – history (date of design,updates, release status, revisions).

E.3.7 Industry segments in which volume based design is applied

Below is a list of products which can be designed very effectively with solid modelling. This list is compiled in industry sectors.

Machinery industry:

This industry manufactures and assembles manufacturing machines for different target markets.

EXAMPLE 14 – Companies manufacturing machines for:

- metal cutting (e.g., lathes, milling machines, grinding machines);
- metal forming (e.g., presses and dies);
- assembly lines (e.g., robots);
- plastic moulding (e.g., plastic injection machines);
- welding constructions (e.g. welding robots);

Electro-mechanical manufacturing industry:

This industry manufactures and assembles electro-mechanical equipment for industrial use and for the consumer market.

EXAMPLE 15 – Companies in this industry manufacture:

- computer products (printers, disc drives, plotters), (parts examples: plastic housings, switches);
- electromechanical equipment for the automotive industry, (electric motors, electricity generator, starter motor, batteries);
- consumer products (e.g. telephones,hand held drilling machines, hair dryers).

Transportation Industry

This industry manufactures and assembles parts used for transportation.

EXAMPLE 16 – Companies which manufacture:

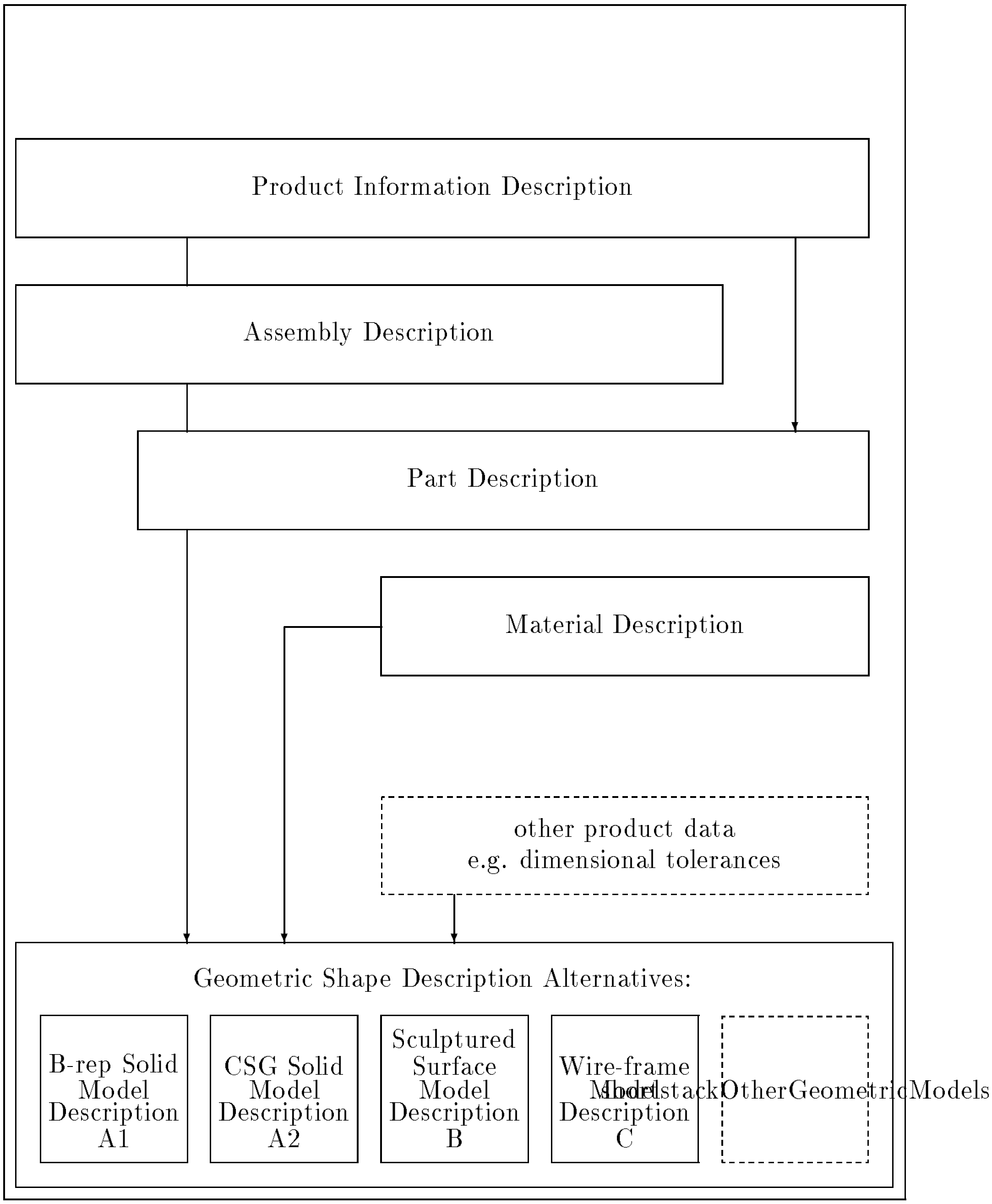

Figure E.1 – conceptual structure of mechanical design product

- automobile engines;
- mechanical power transmission equipment, gears, transaxles;
- wheels, tyres;
- hydraulic equipment e.g., valves;

E.4 AAM Diagrams

The application activity model is given in E.1 through E.4. The graphical form of the application activity model is presented in the IDEF0 activity modeling format. Activities and data flows that are out of scope are marked with asterisks.

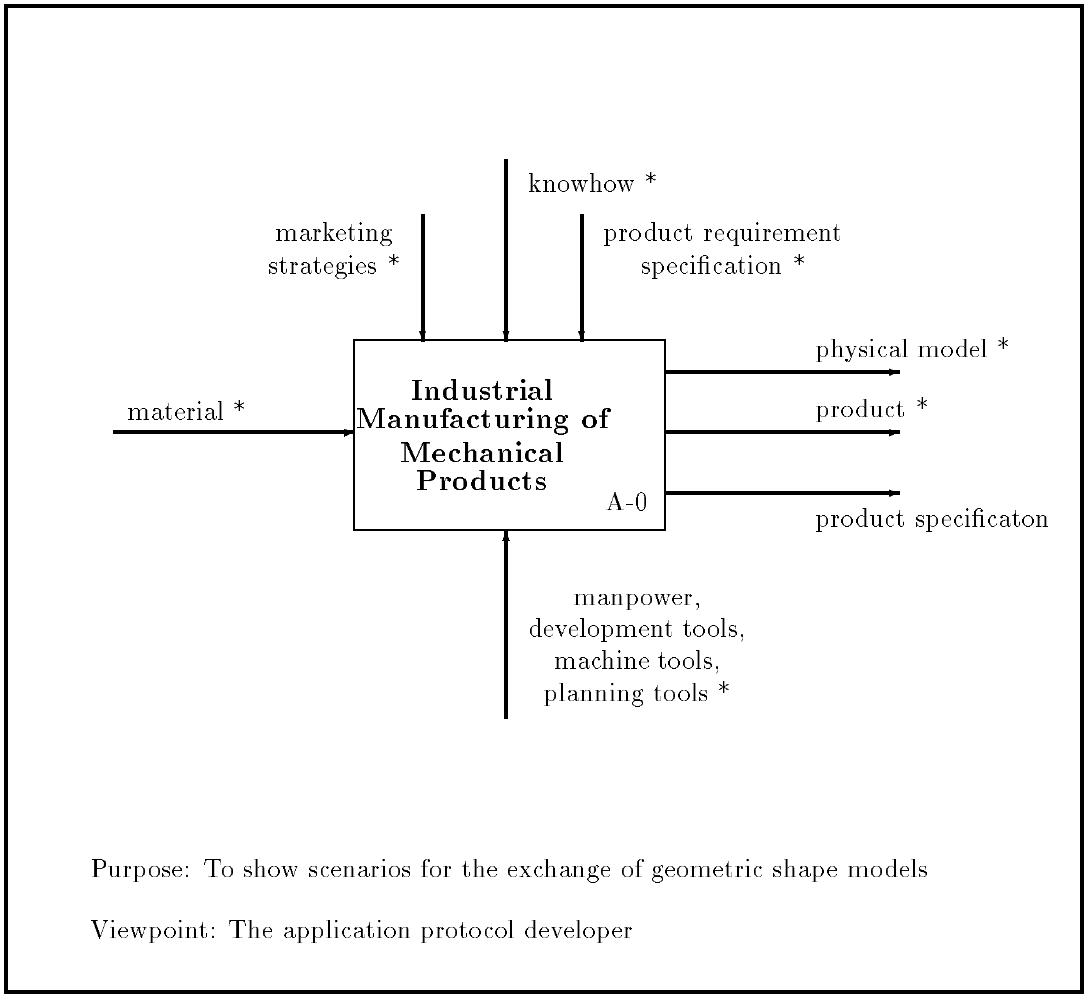

Figure E.2 – Industrial manufacturing of mechanical products (node A0)

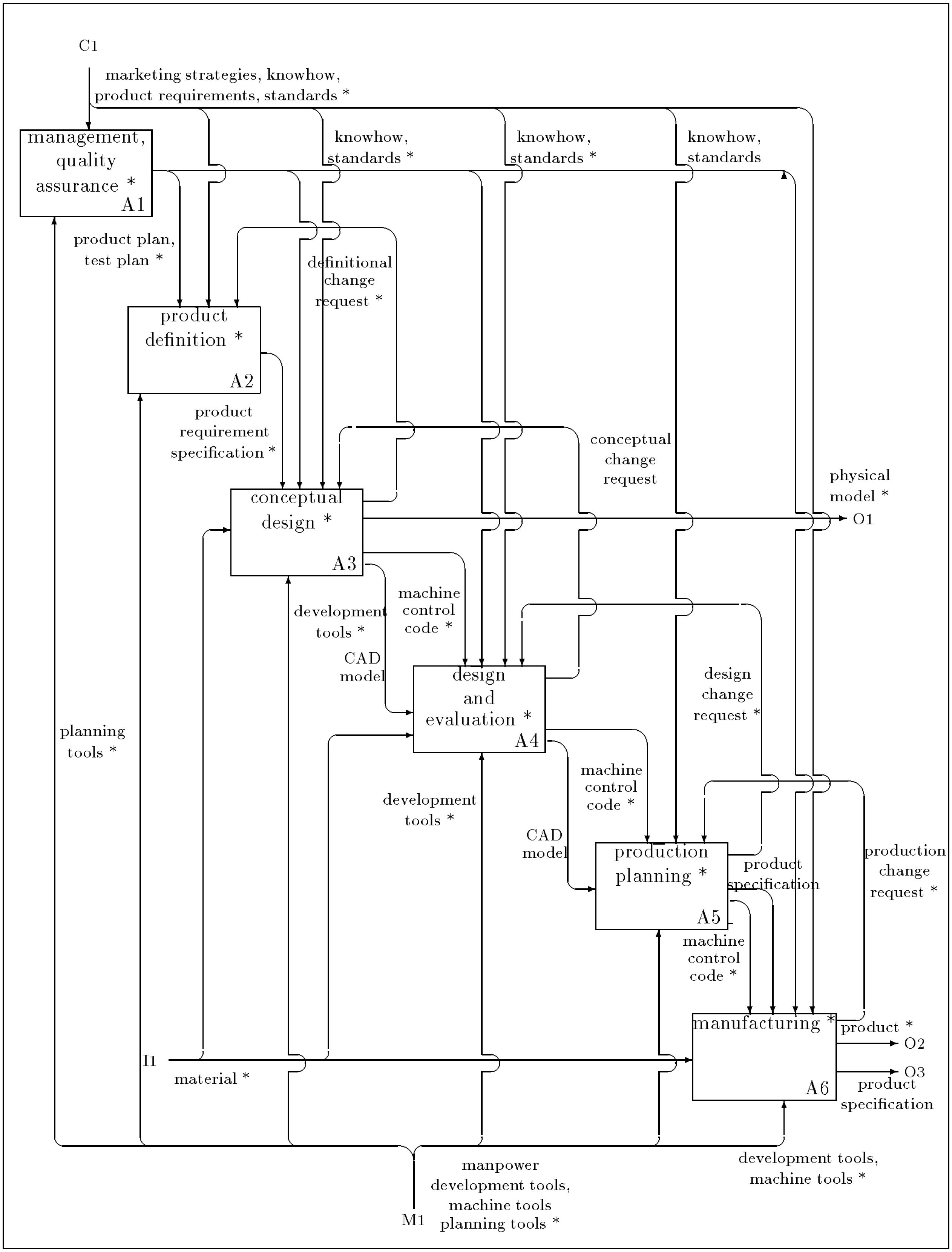

Figure E.3 – Industrial manufacturing of mechanical products (node A0 expanded)

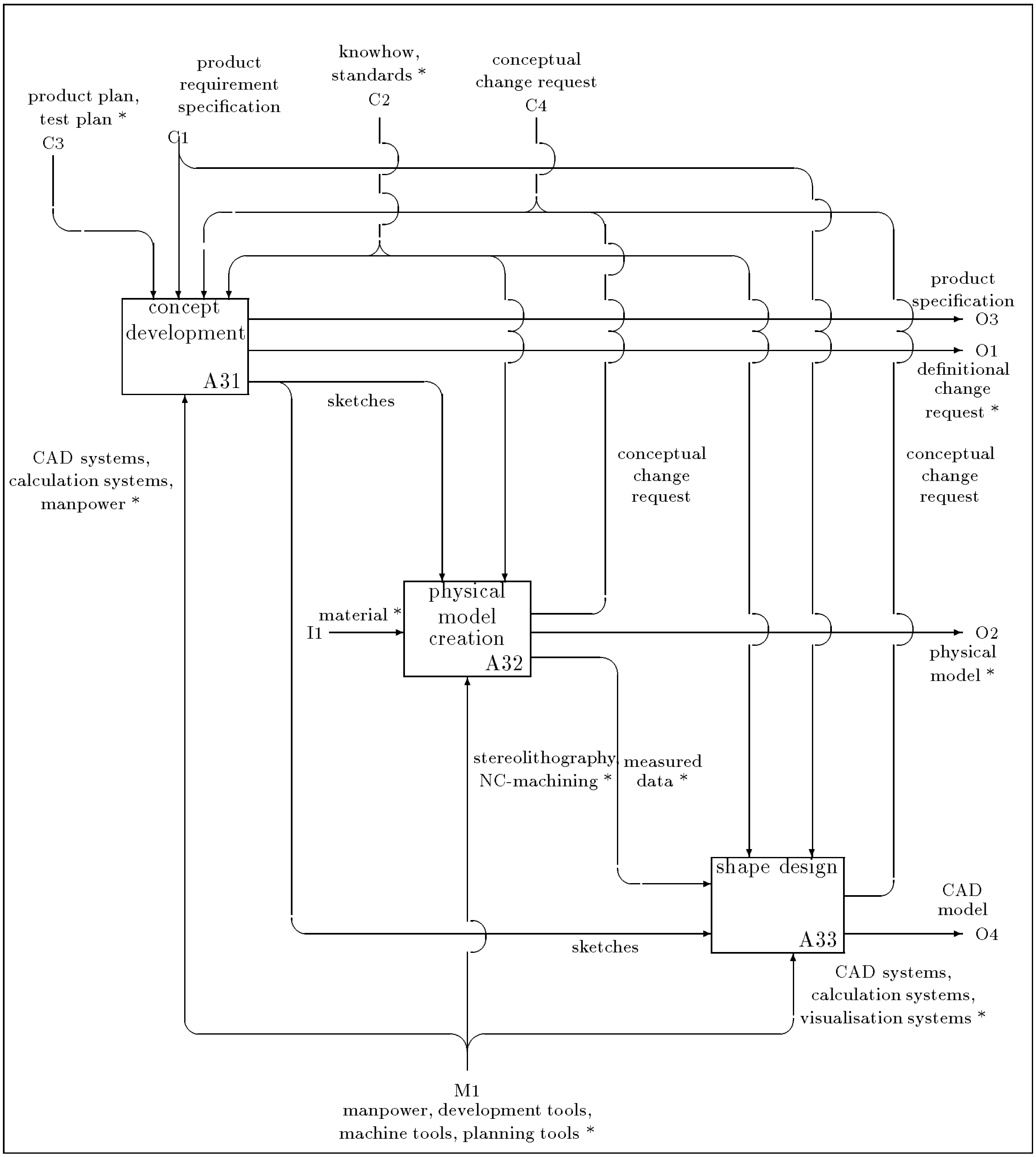

Figure E.4 – Conceptual design (node A3)

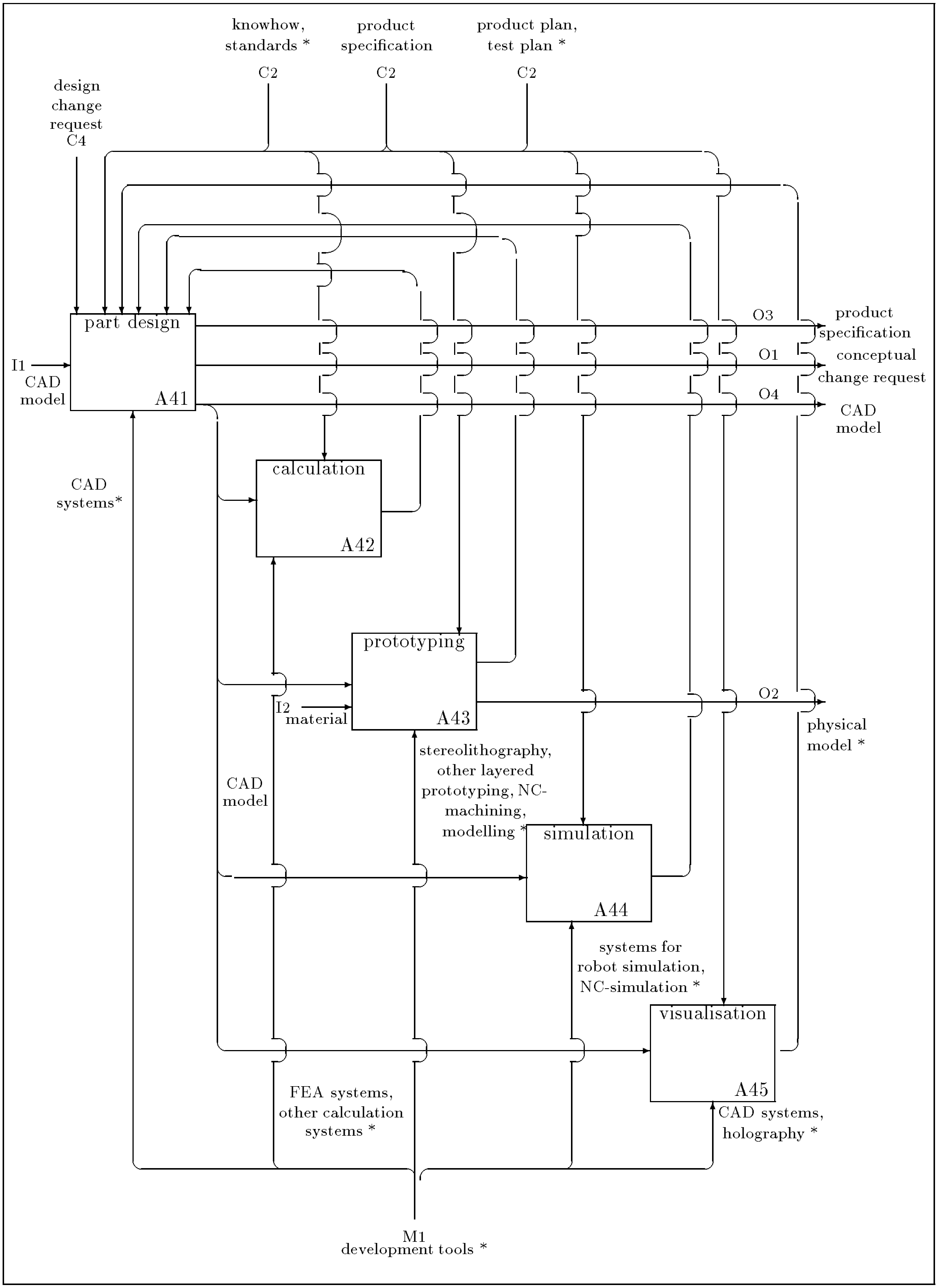

Figure E.5 – Design and evaluation (Node A4)

Annex F
(informative)

Application Reference Model Diagrams

This annex provides the application reference model for this part of ISO 10303 and is given in F.1 through F.11. The application reference model is a graphical representation of the structure and constraints of the application objects specified in clause 4. The graphical form of the application reference model is presented in the NIAM format. The application reference model is independent from any implementation method.

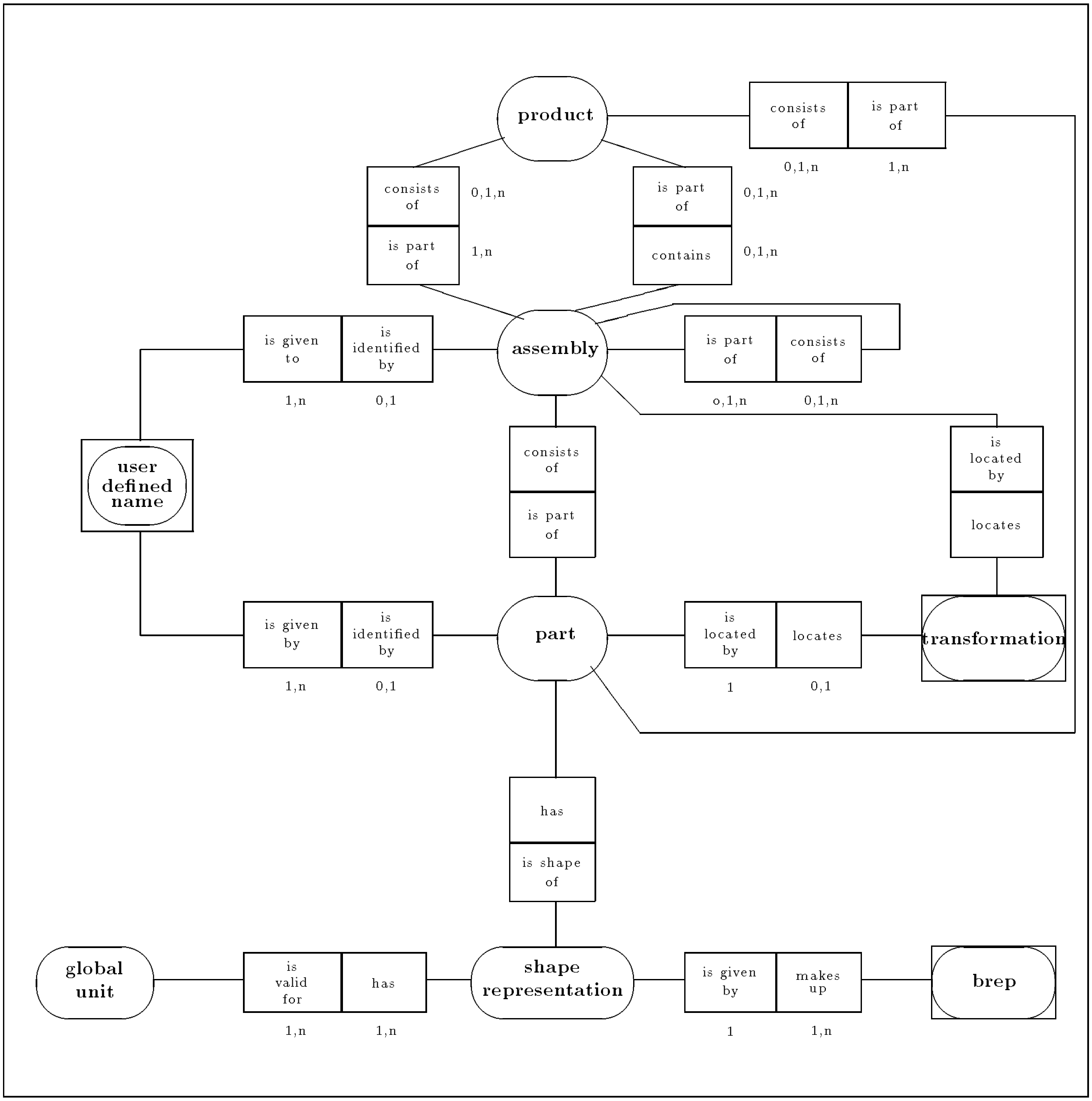

Figure F.1 – Product, assembly and part structure

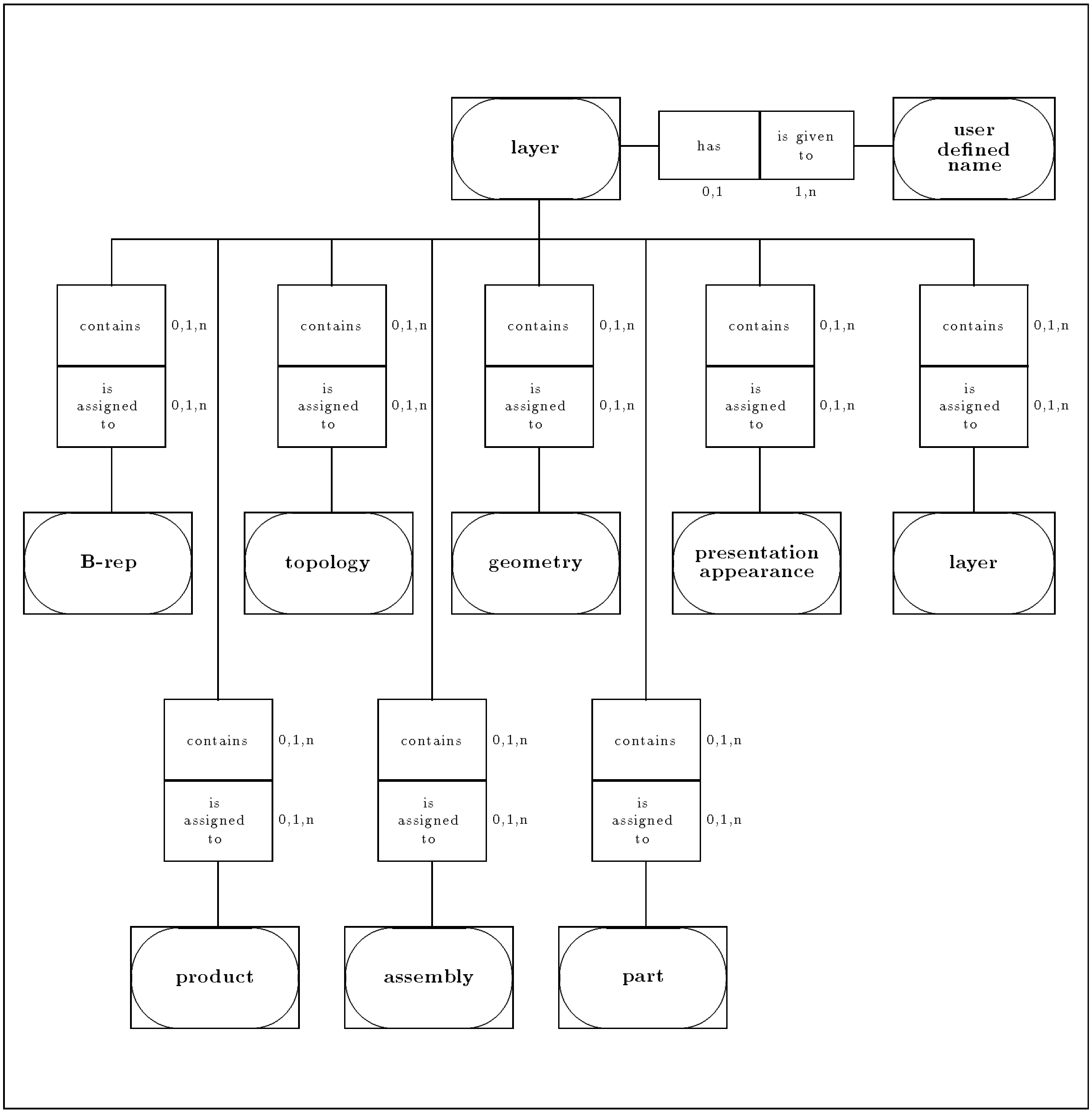

Figure F.2 – Layer structure and presentation attributes

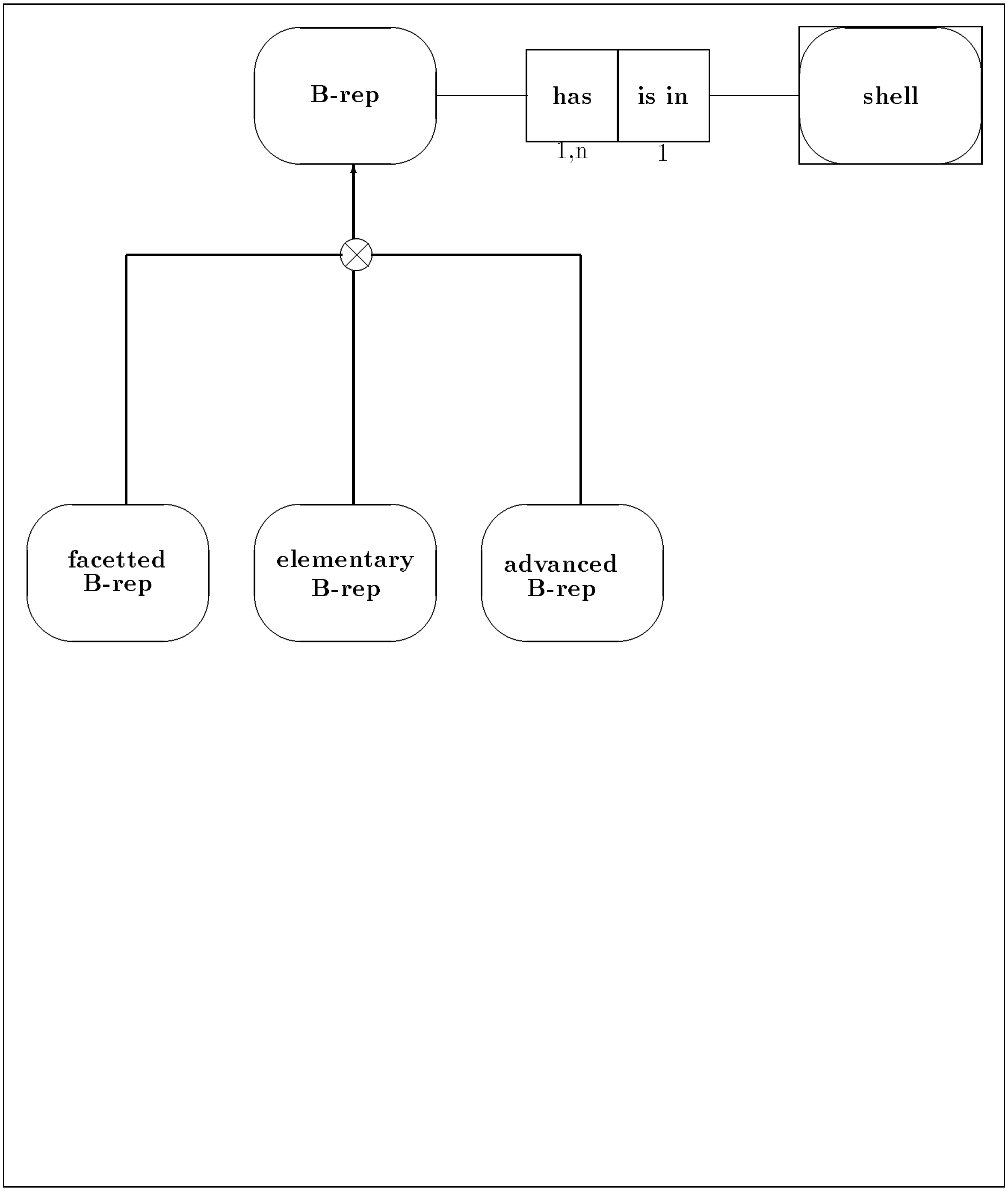

Figure F.3 – B-rep

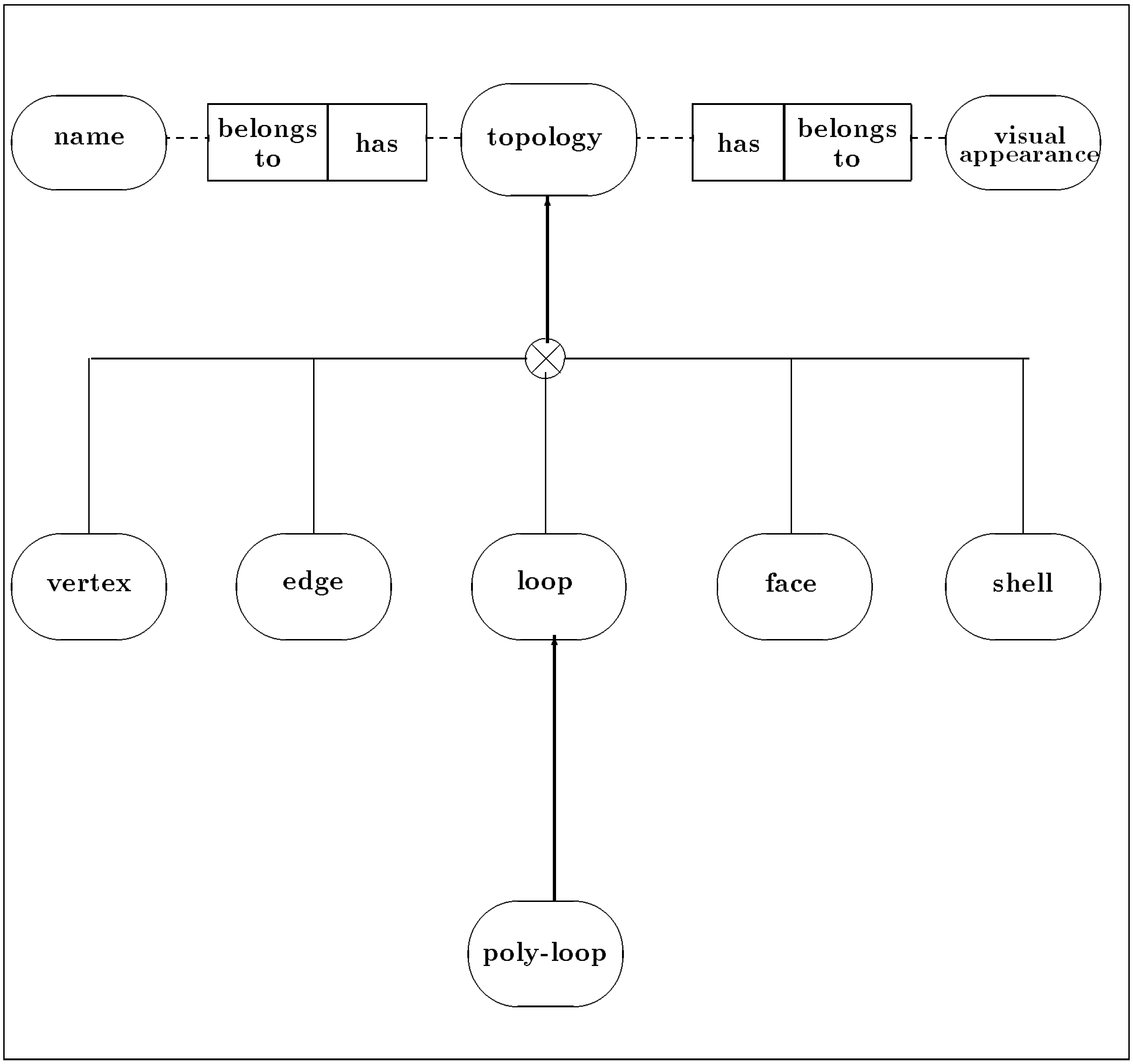

Figure F.4 – Topology elements

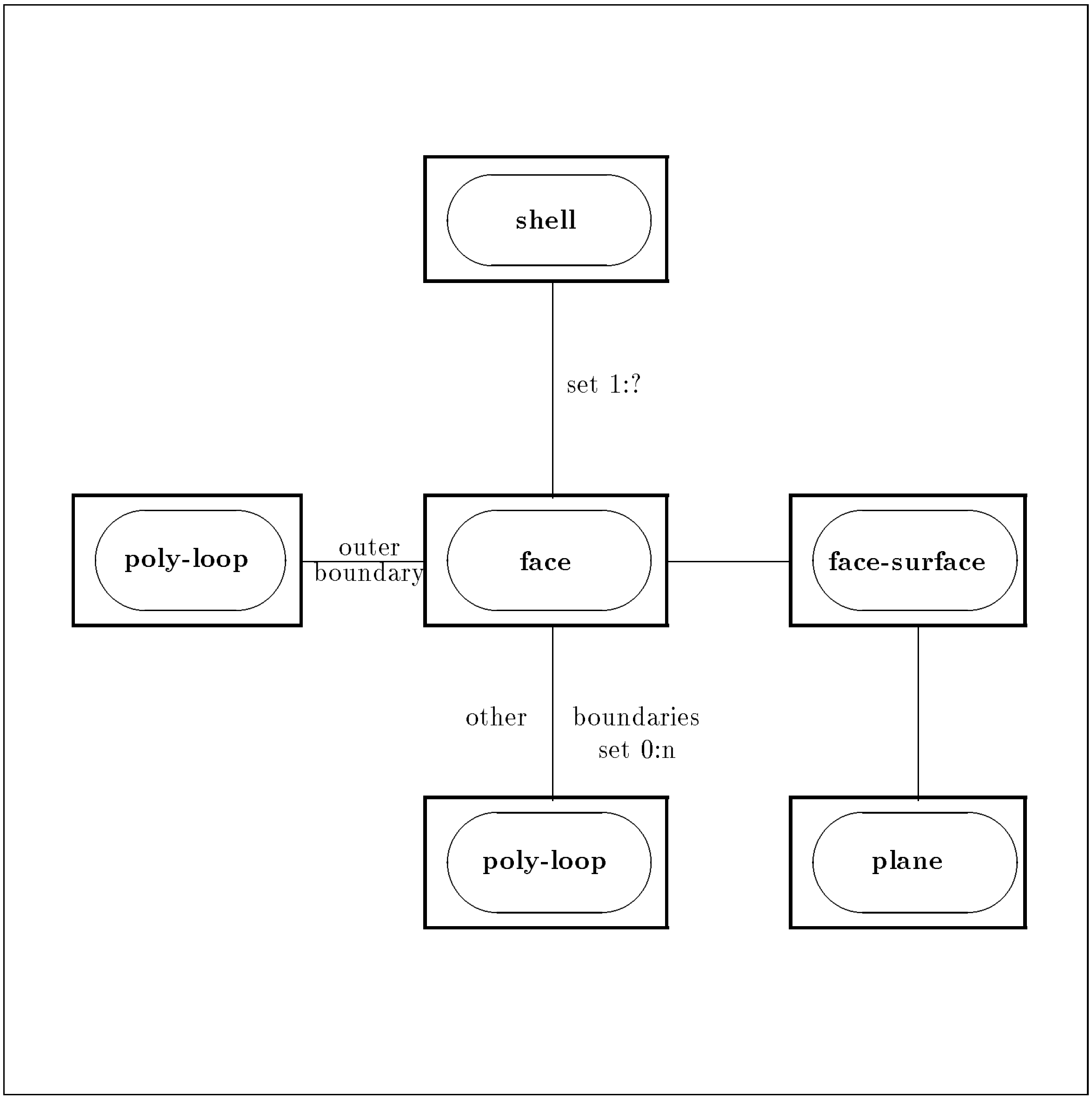

Figure F.5 – Shell in Facetted B-rep

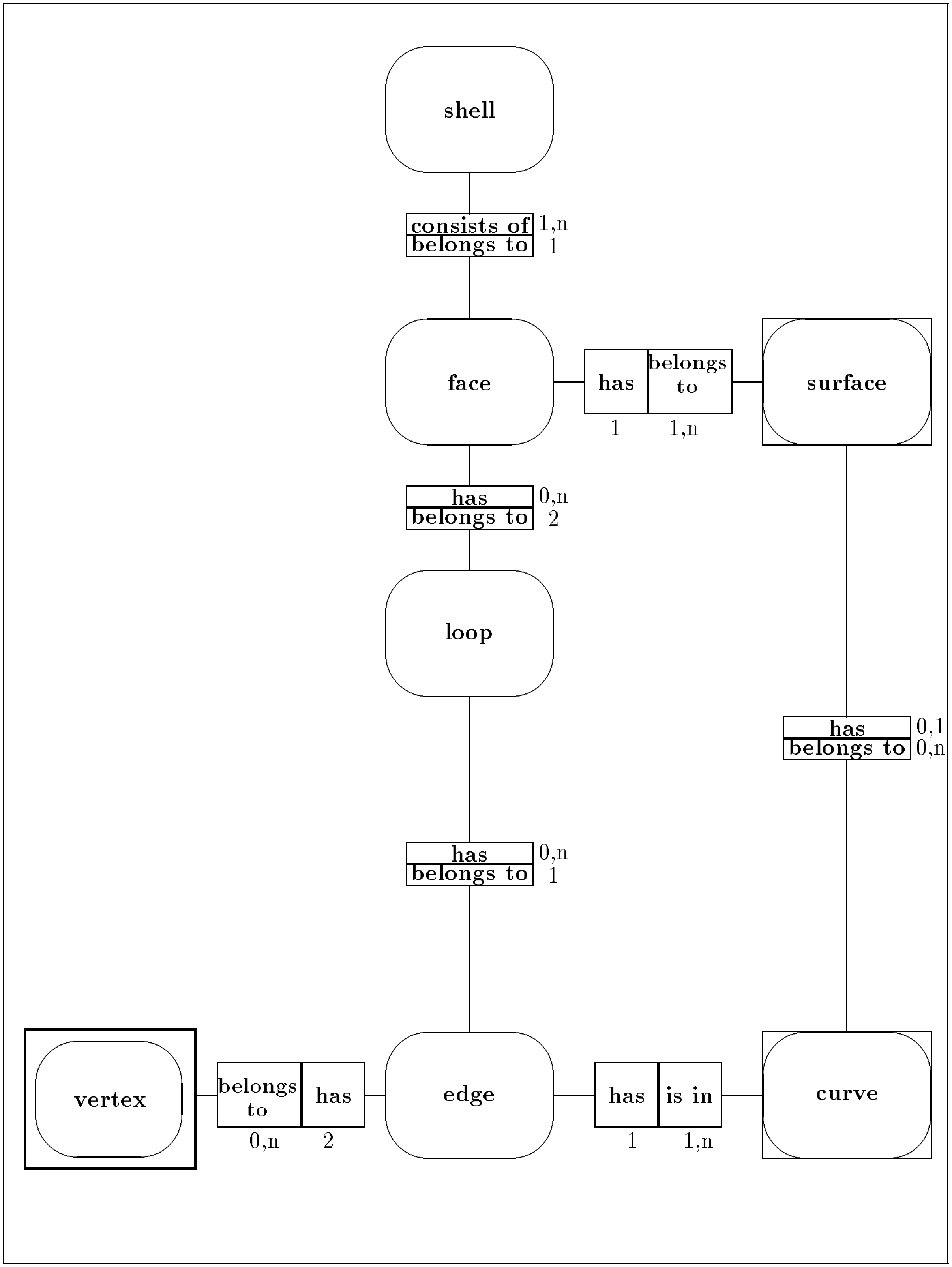

Figure F.6 – Shell in Elementary or Advanced B-rep

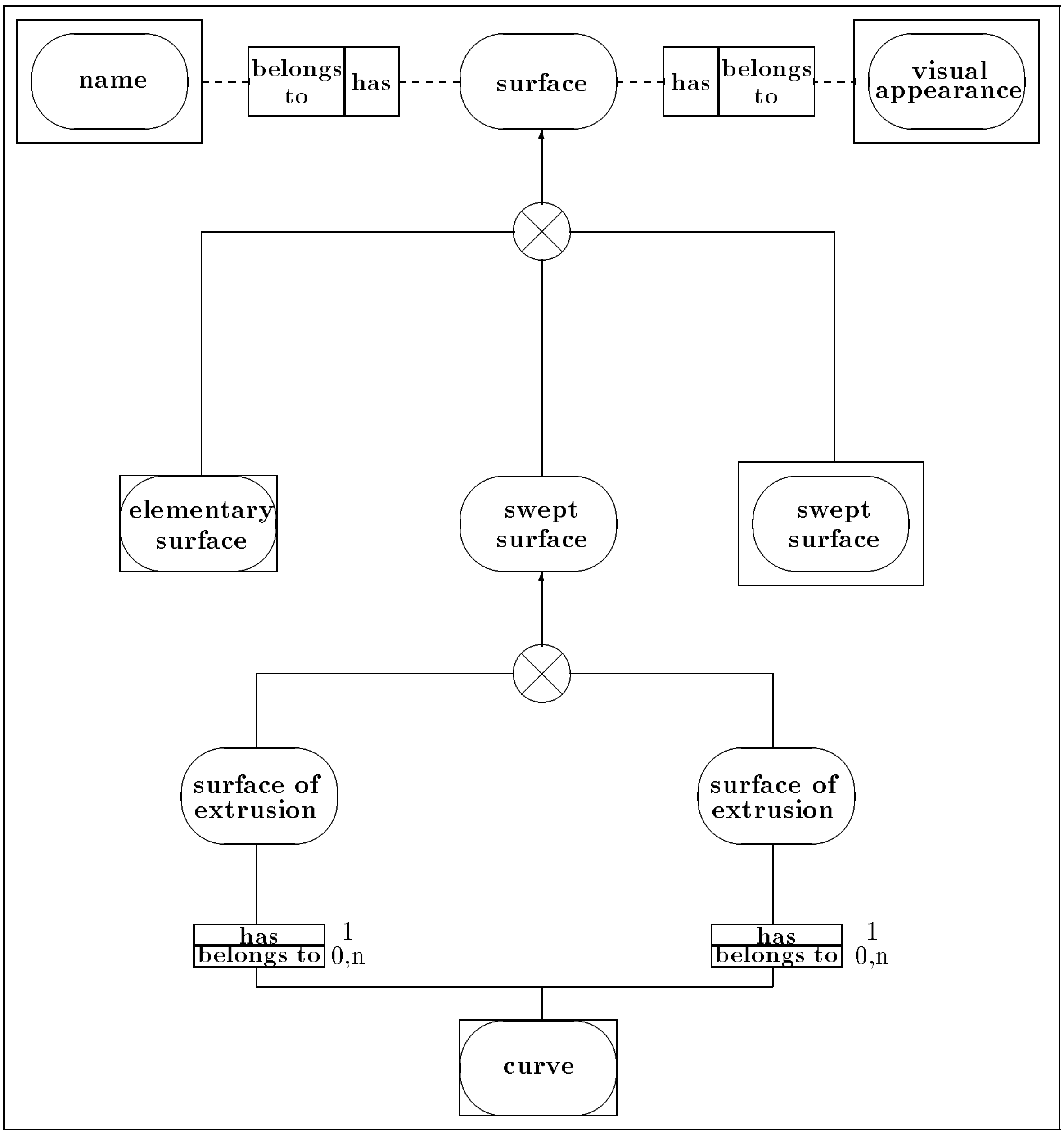

Figure F.7 – Surface in Advanced B-rep

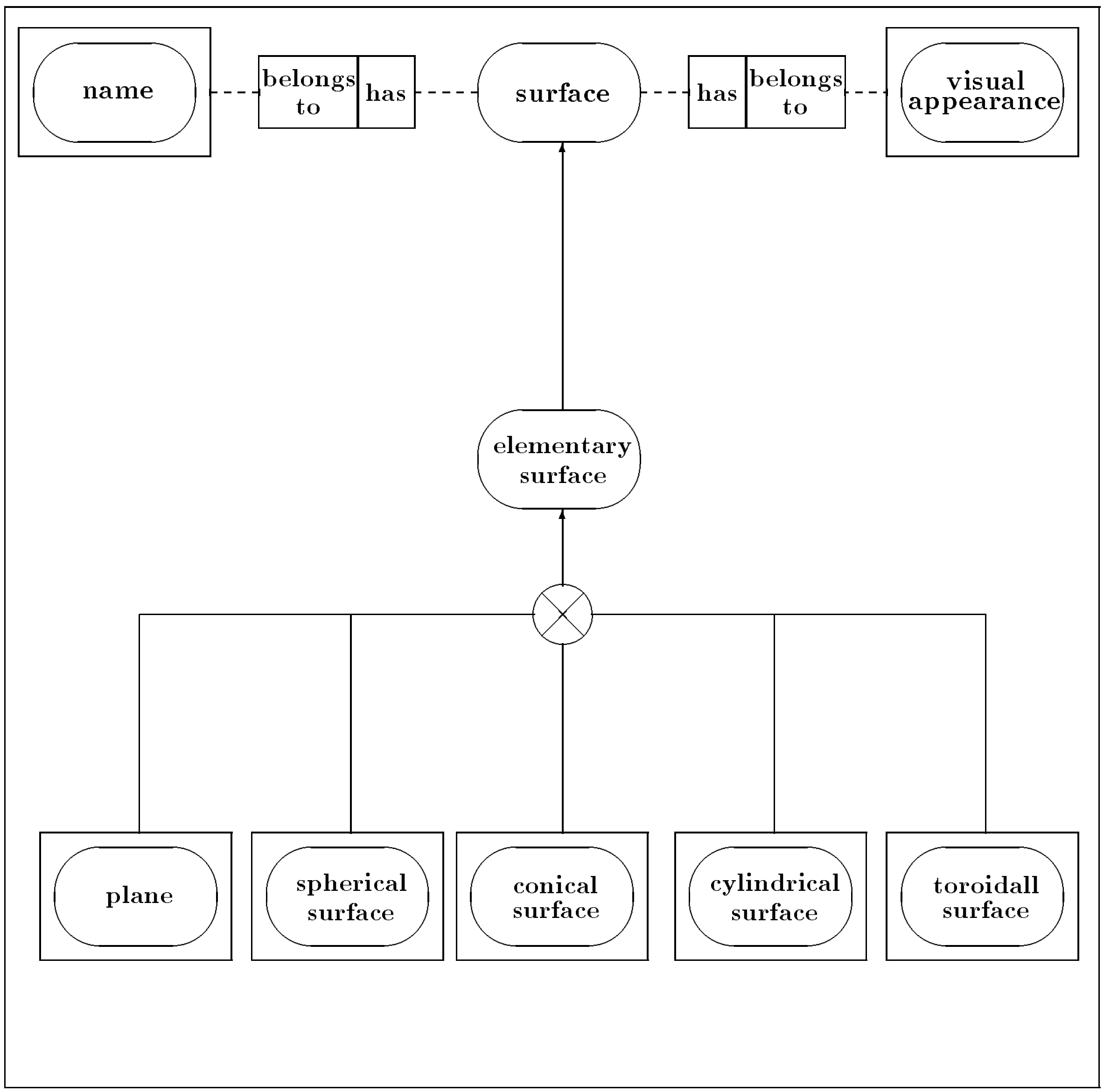

Figure F.8 – Surface in Elementary B-rep

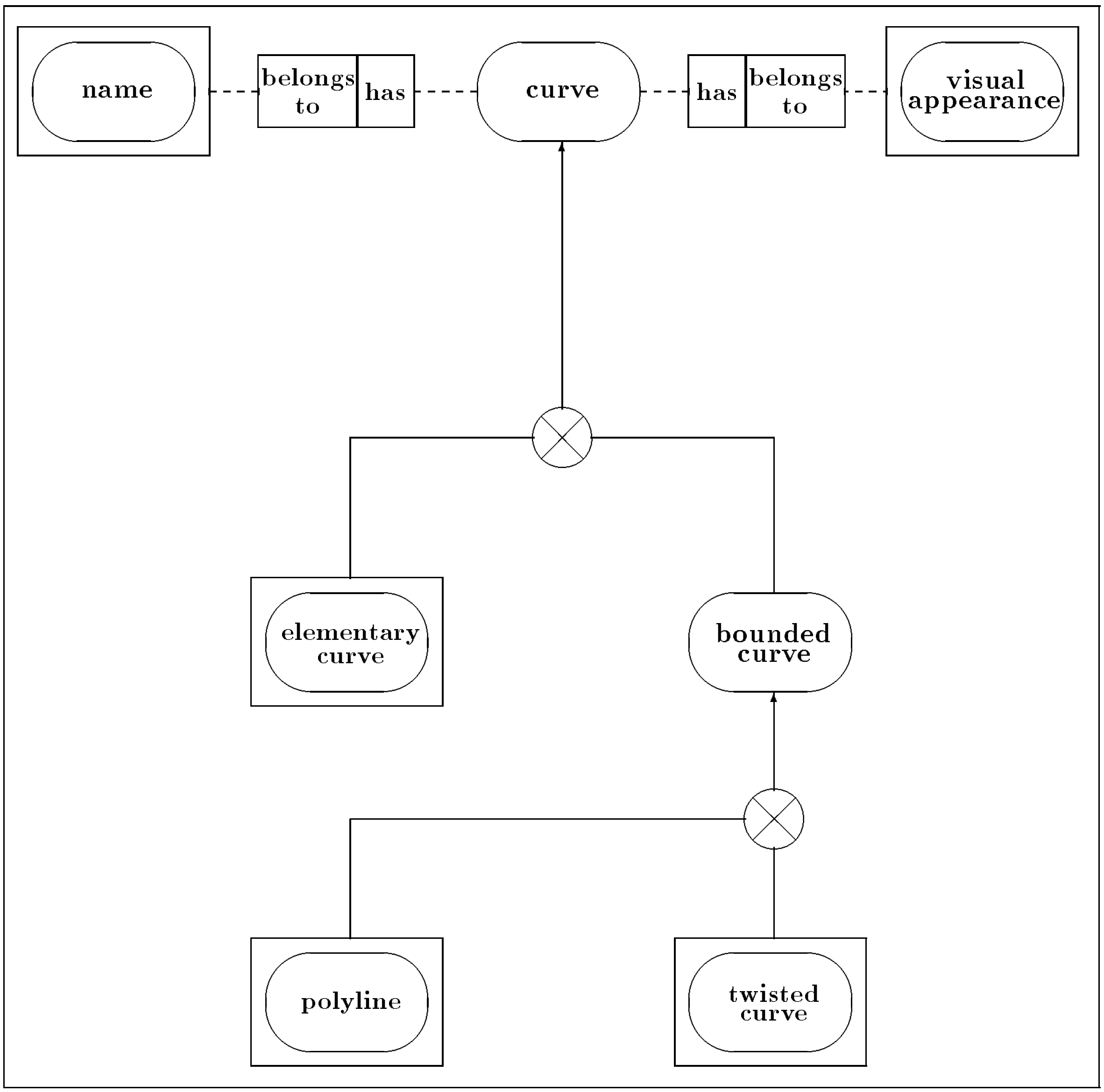

Figure F.9 – Curve in Advanced B-rep

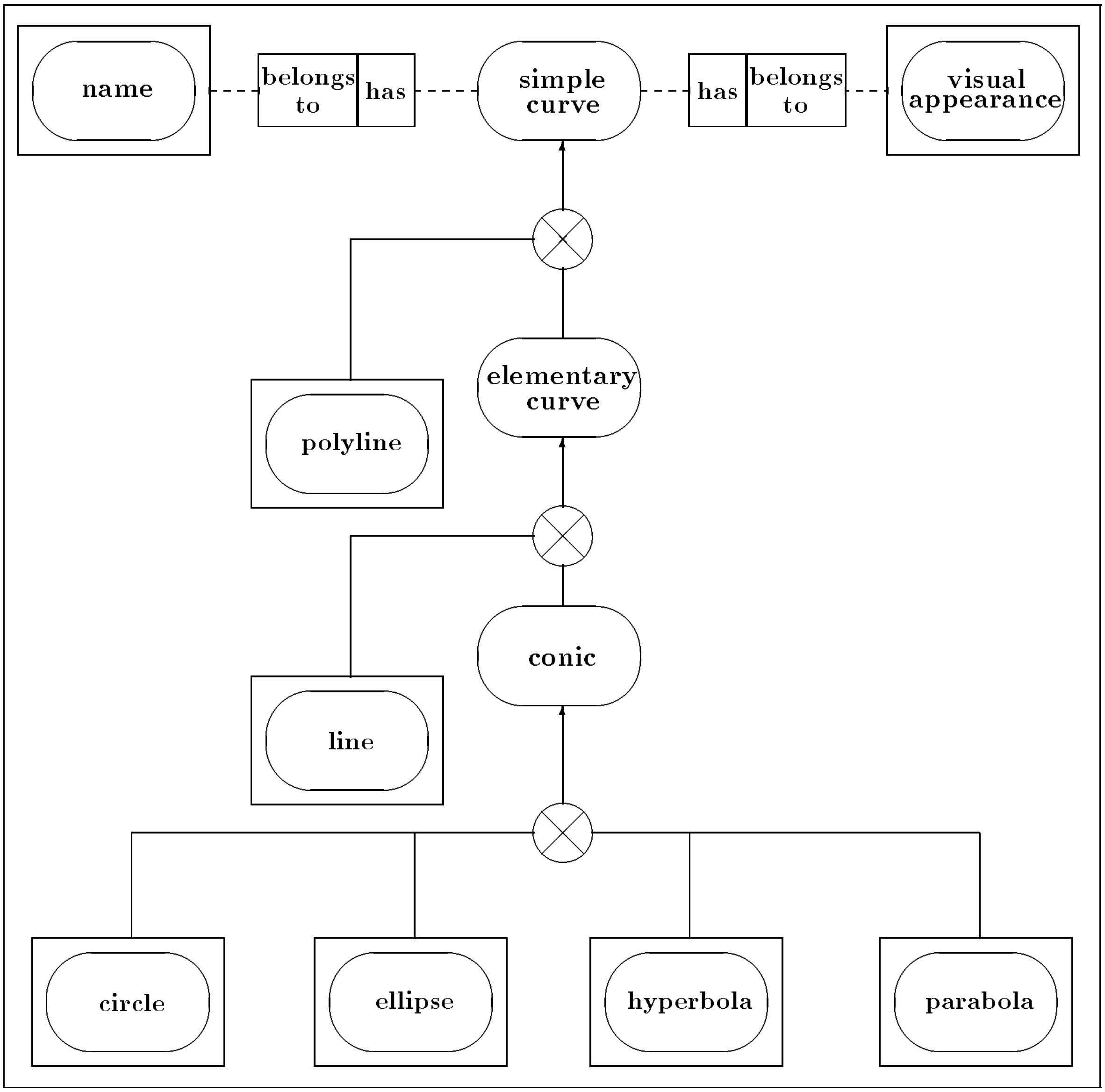

Figure F.10 – Curve in Elementary B-rep

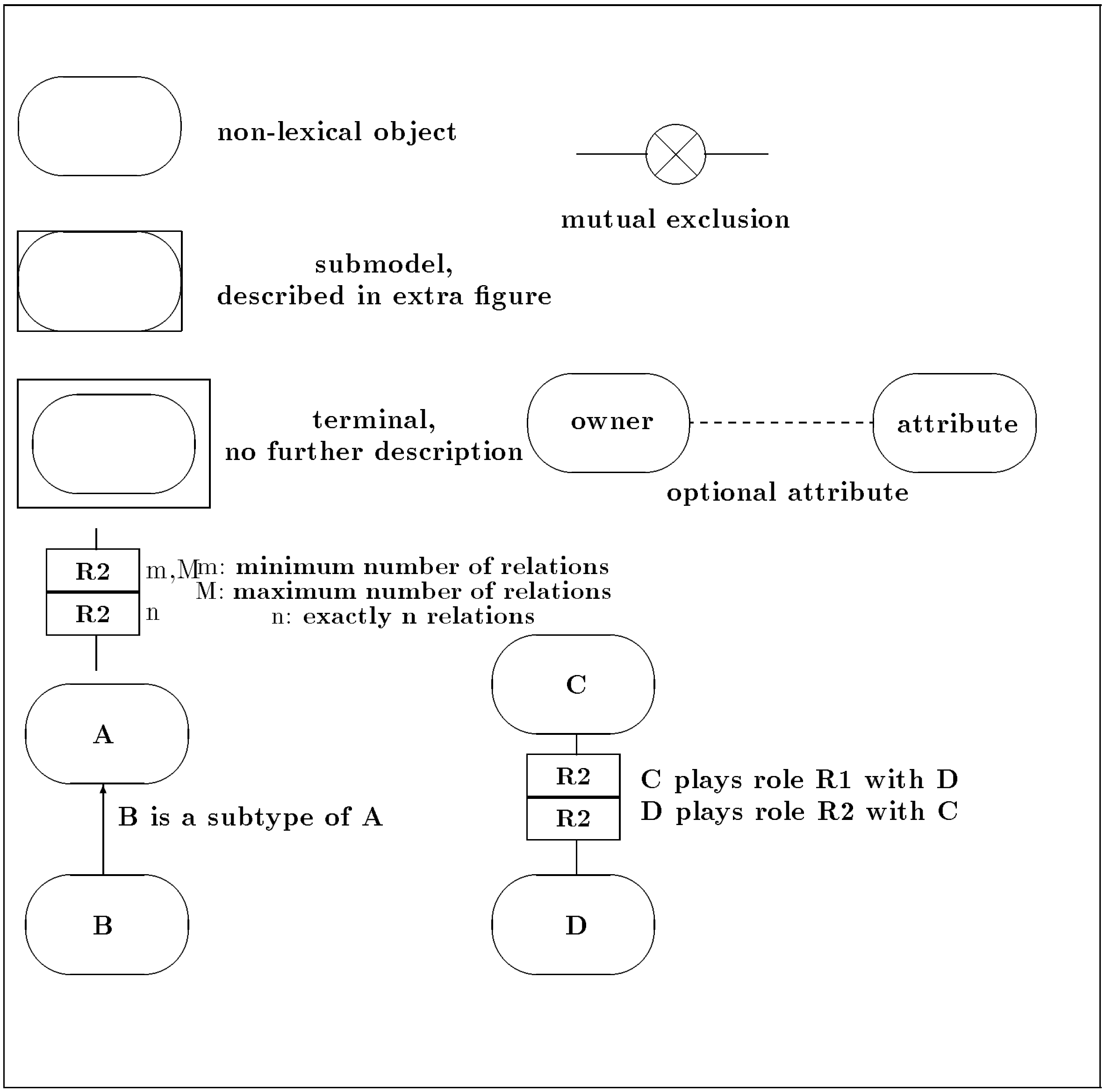

Figure F.11 – Conventions used in NIAM diagrams

Annex G
(informative)

Application Interpreted Model Diagrams

Figures G.1 through G.17 correspond to the AIM *EXPRESS* annotated listing given in annex A. The figures use the *EXPRESS-G* graphical notation for the *EXPRESS* language. *EXPRESS-G* is defined in annex D of ISO 10303-11.

Figures G.1 through G.17 correspond to the AIM EXPRESS annotated listing given in annex A. The figures use the EXPRESS-G graphical notation for the EXPRESS language. EXPRESS-G is defined in annex A of ISO 10303-11.

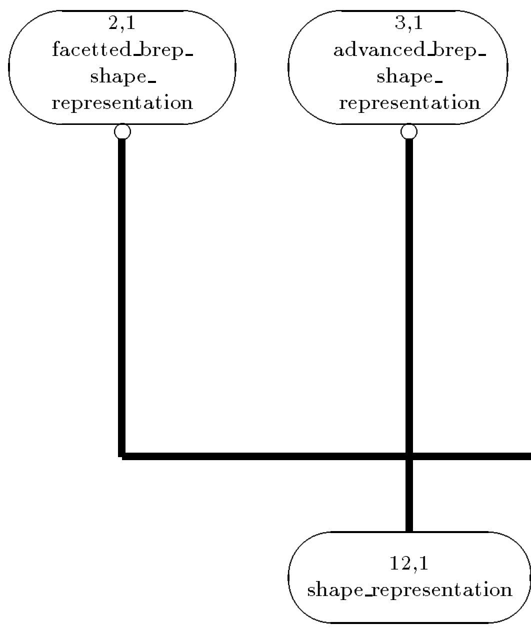

Figure G.1 – Express-G diagram, B-rep models (figure 1 of 17)

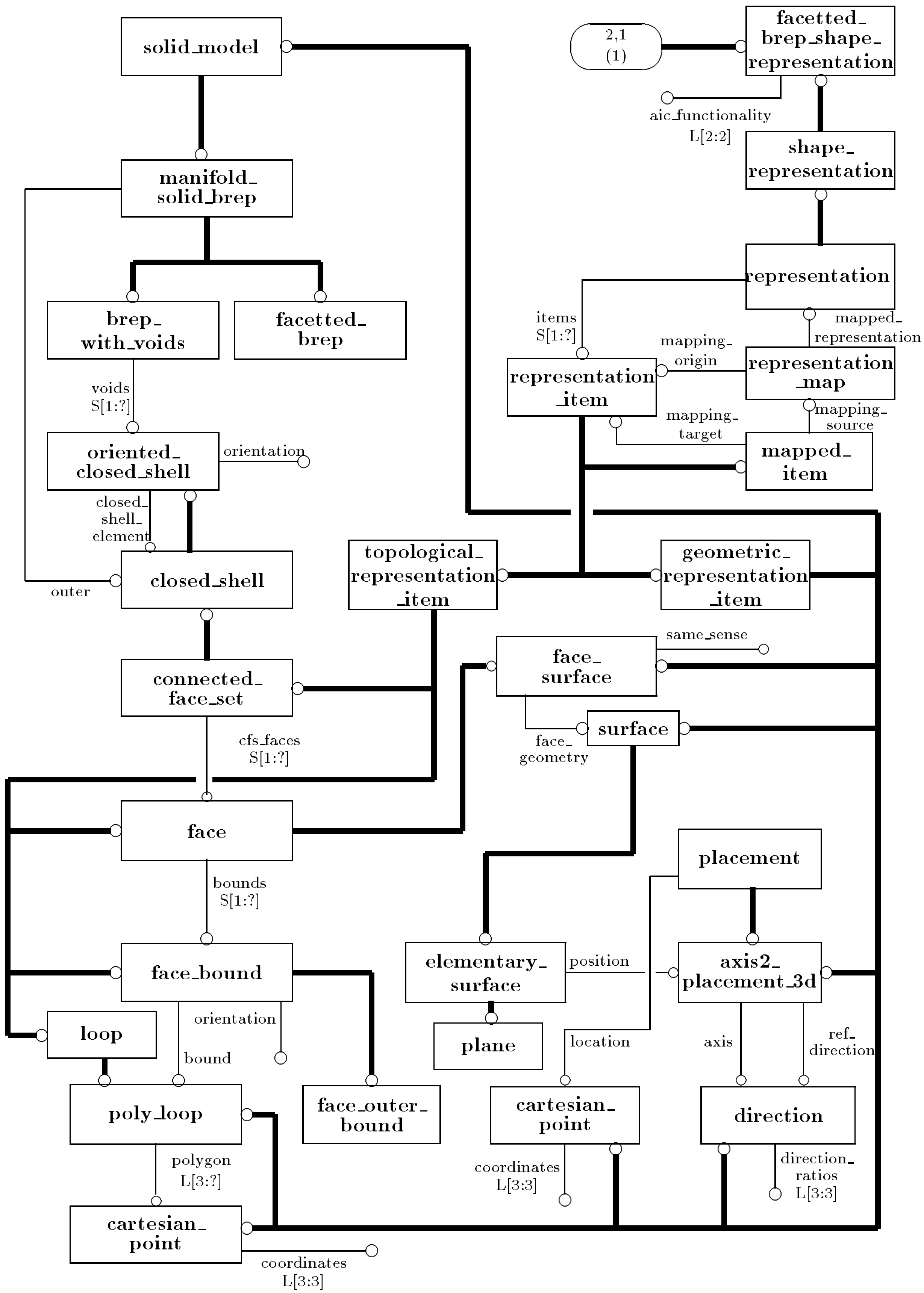

Figure G.2 – Express-G diagram, Facetted B-rep (figure 2 of 17)

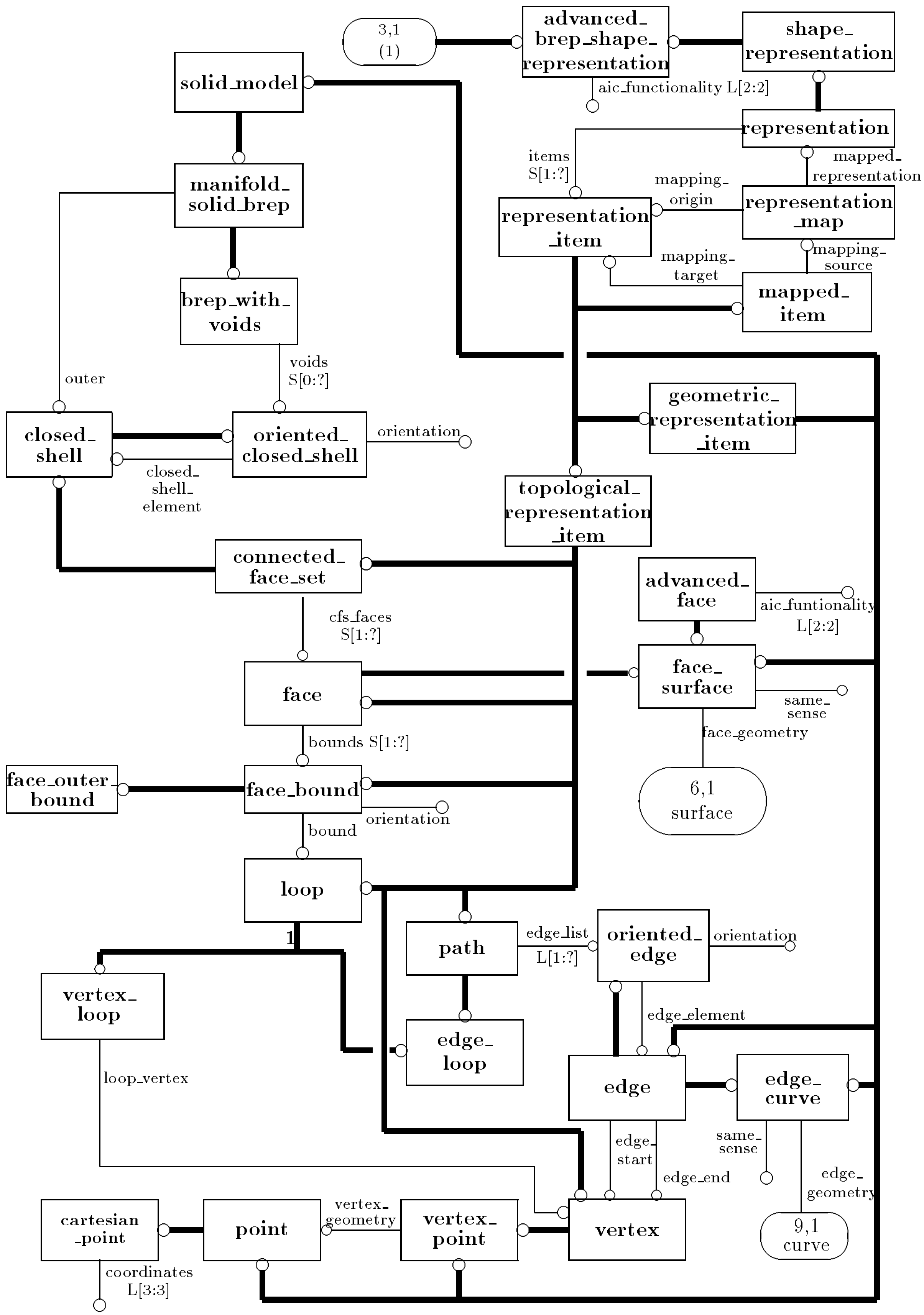

Figure G.3 – Express-G diagram, advanced B-rep model (figure 3 of 17)

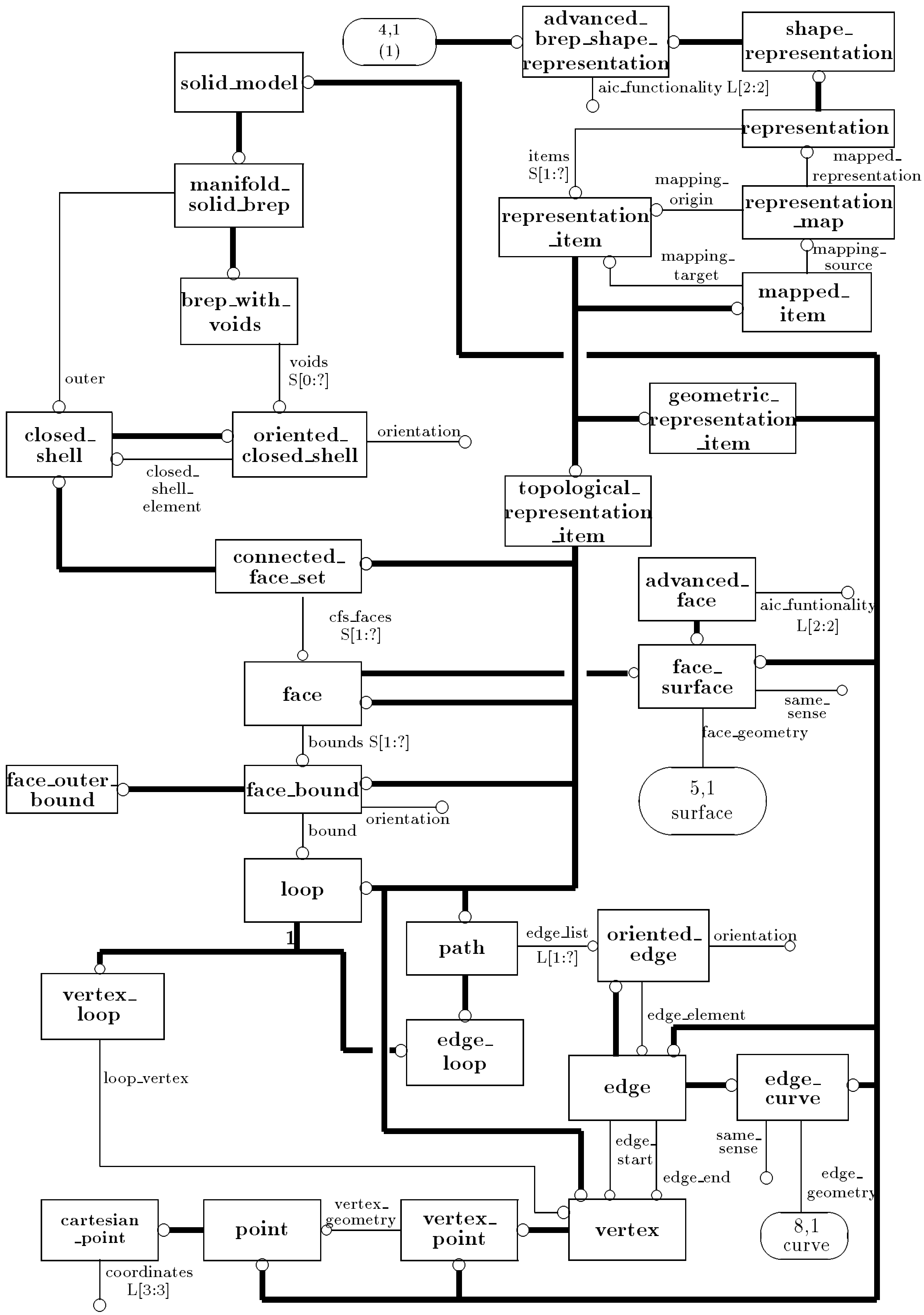

Figure G.4 – Express-G diagram, elementary B-rep model (figure 4 of 17)

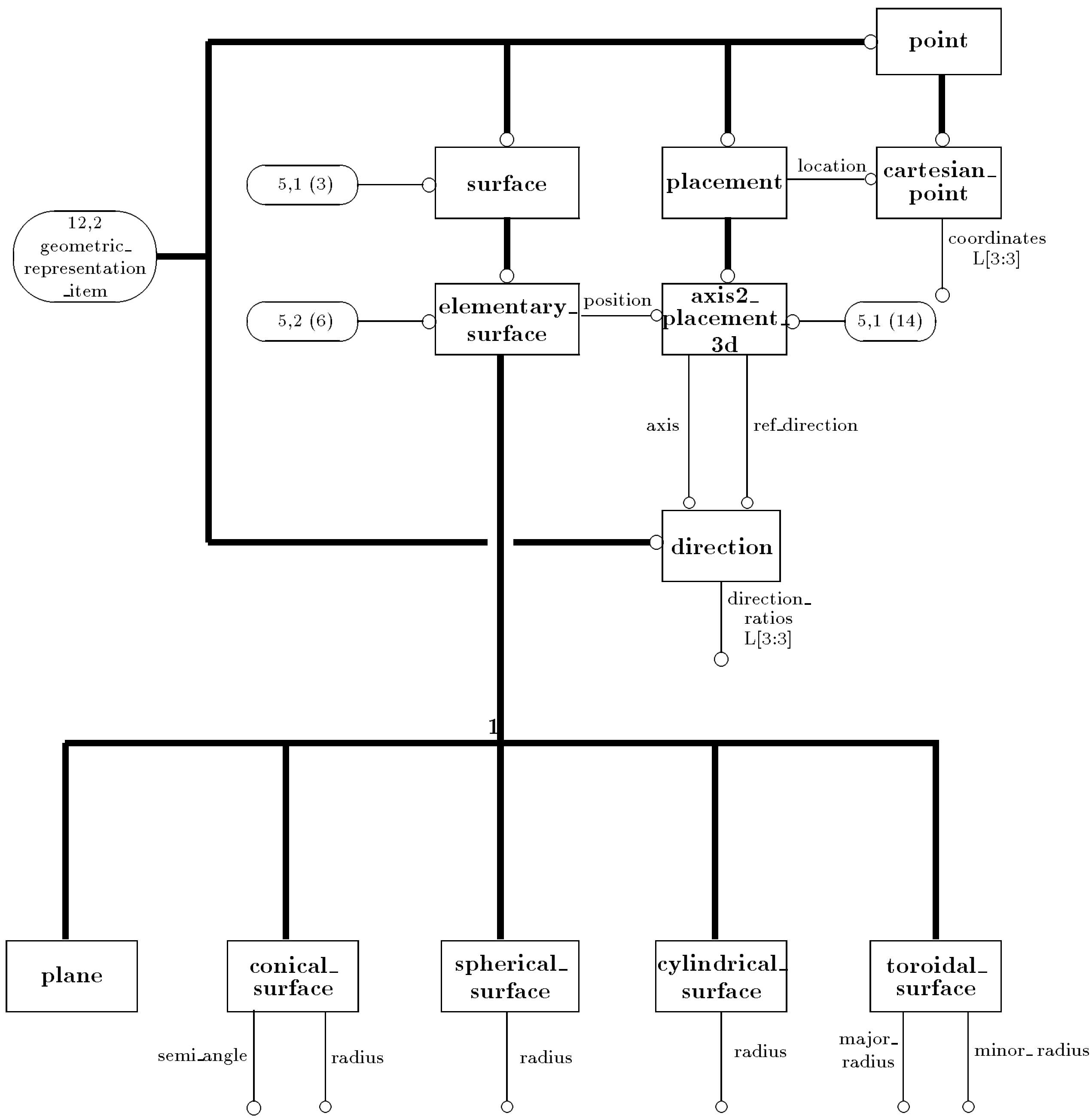

Figure G.5 – Express-G diagram, Surface in Elementary Brep (figure 5 of 17)

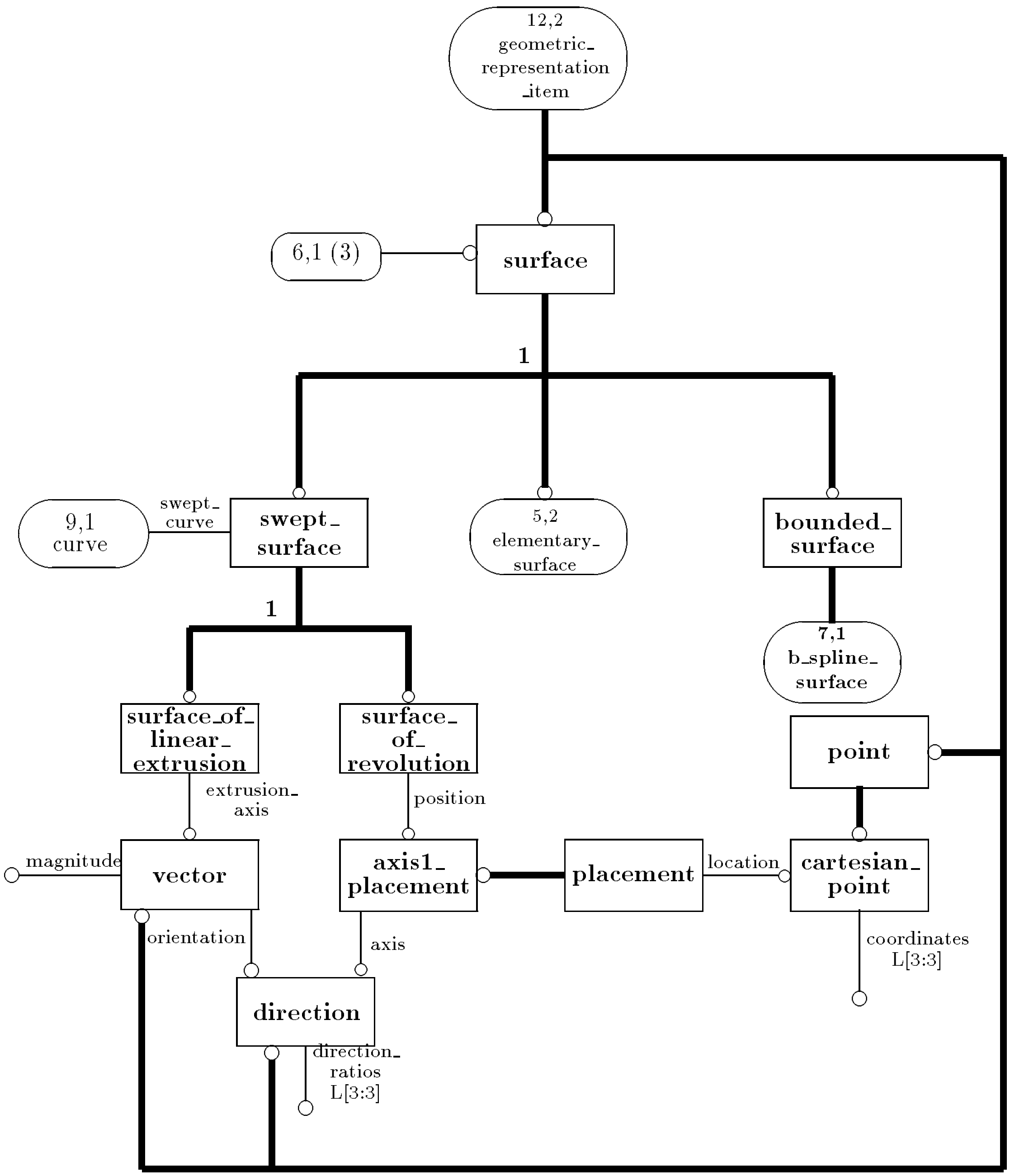

Figure G.6 – Express-G diagram, Surface in Advanced Brep (figure 6 of 17)

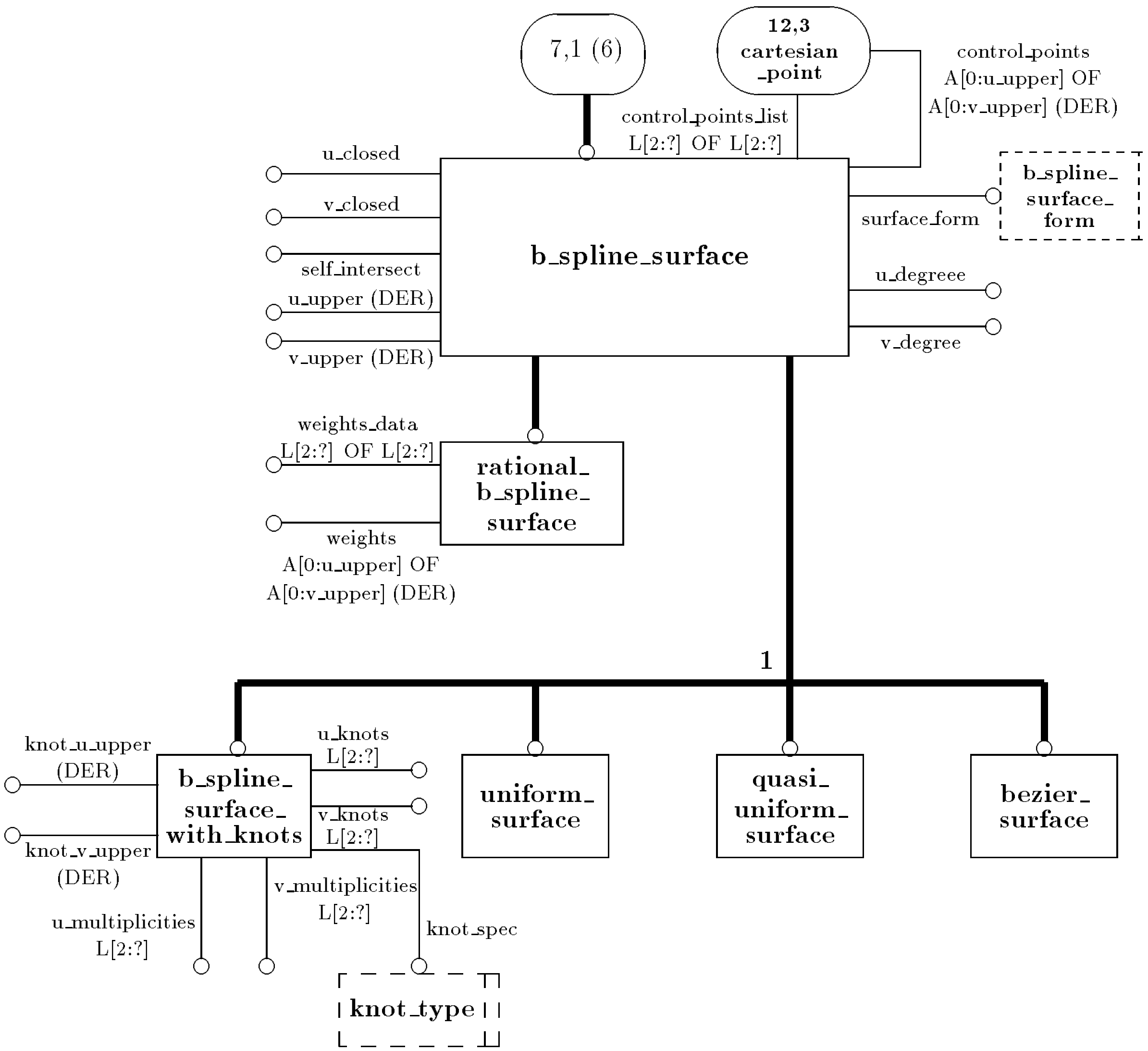

Figure G.7 – Express-G diagram, B_Spline Surface (figure 7 of 17)

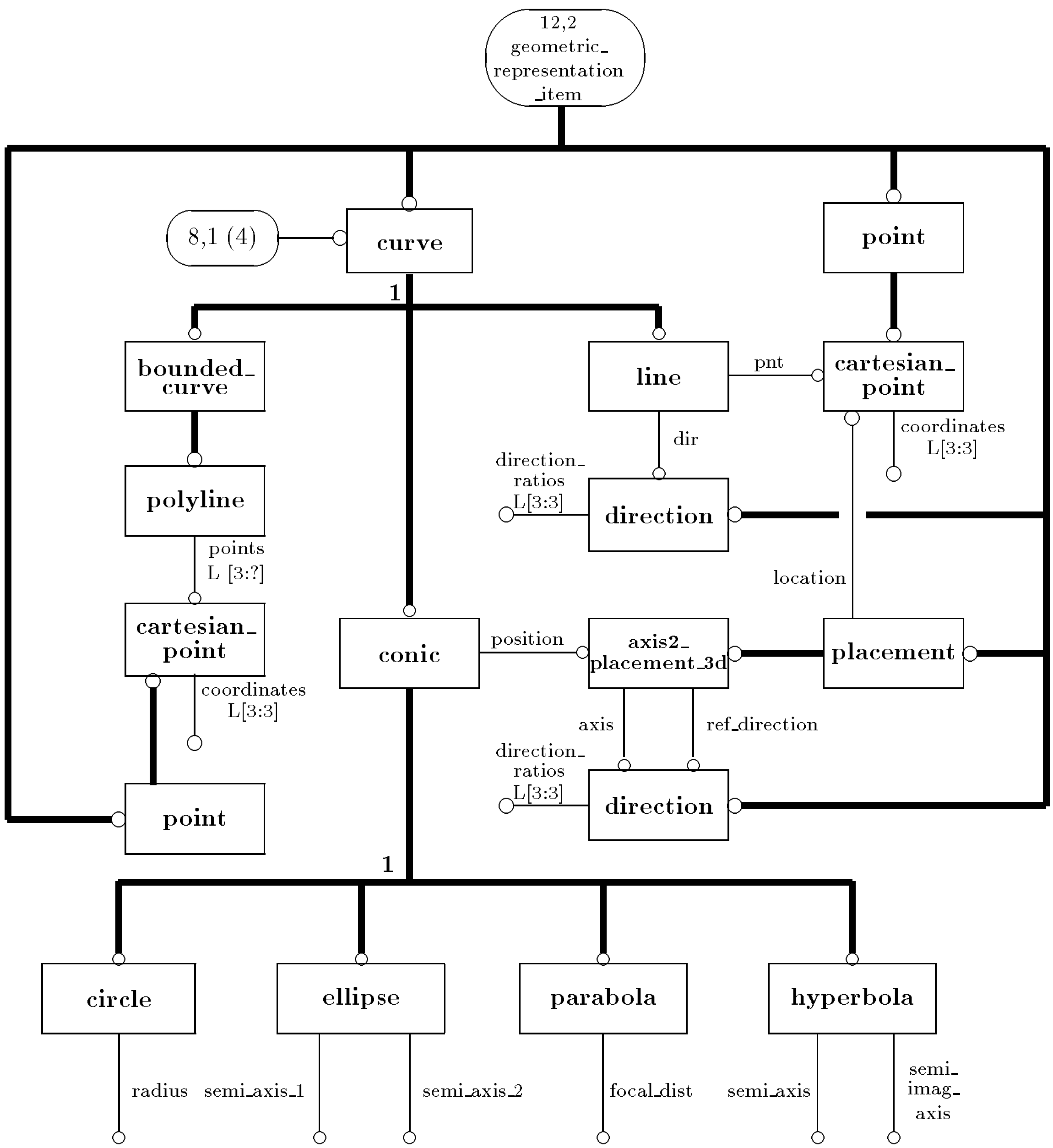

Figure G.8 – Express-G diagram, Curve in Elementary Brep (figure 8 of 17)

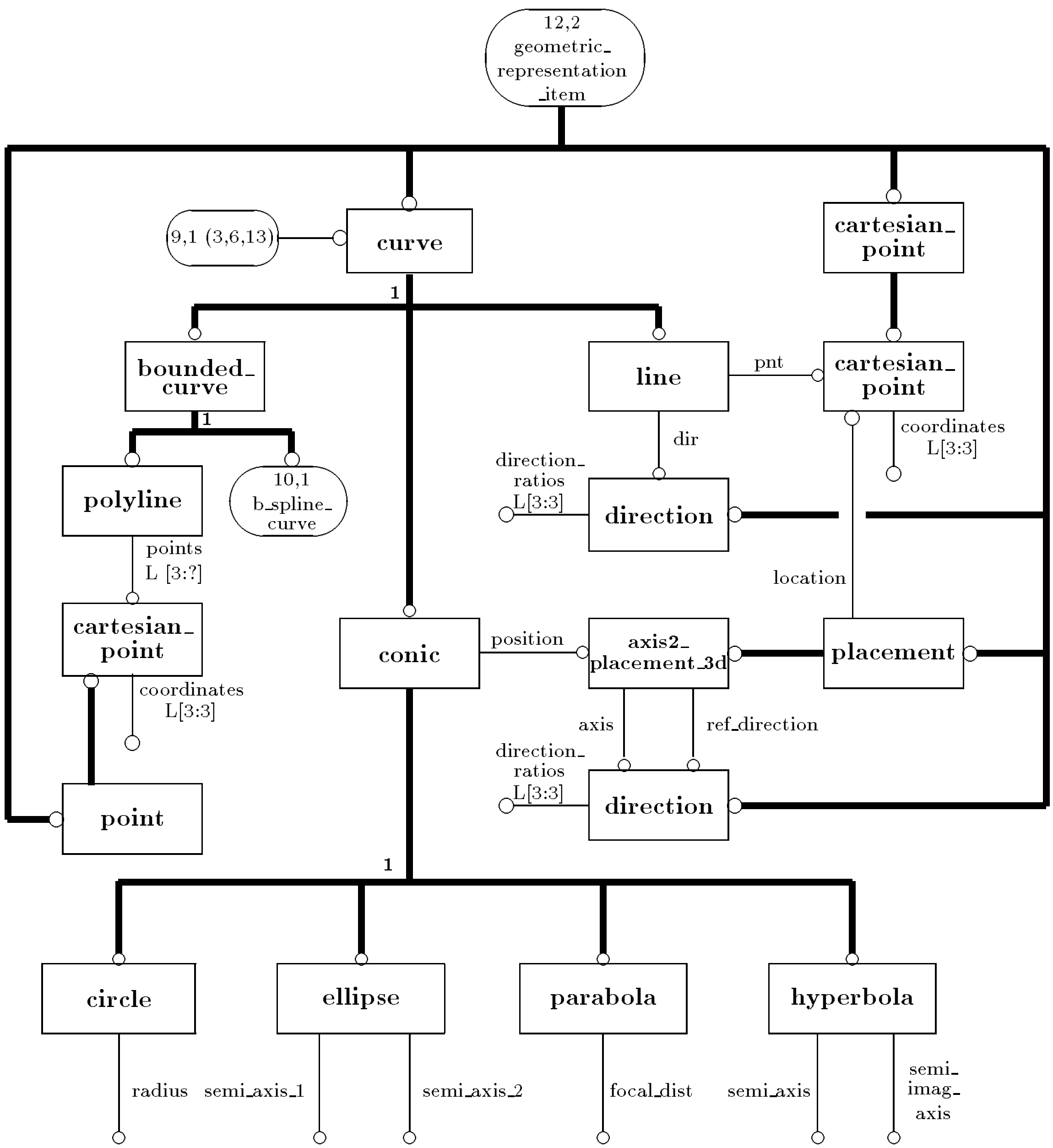

Figure G.9 – Express-G diagram, Curve in Advanced Brep (figure 9 of 17)

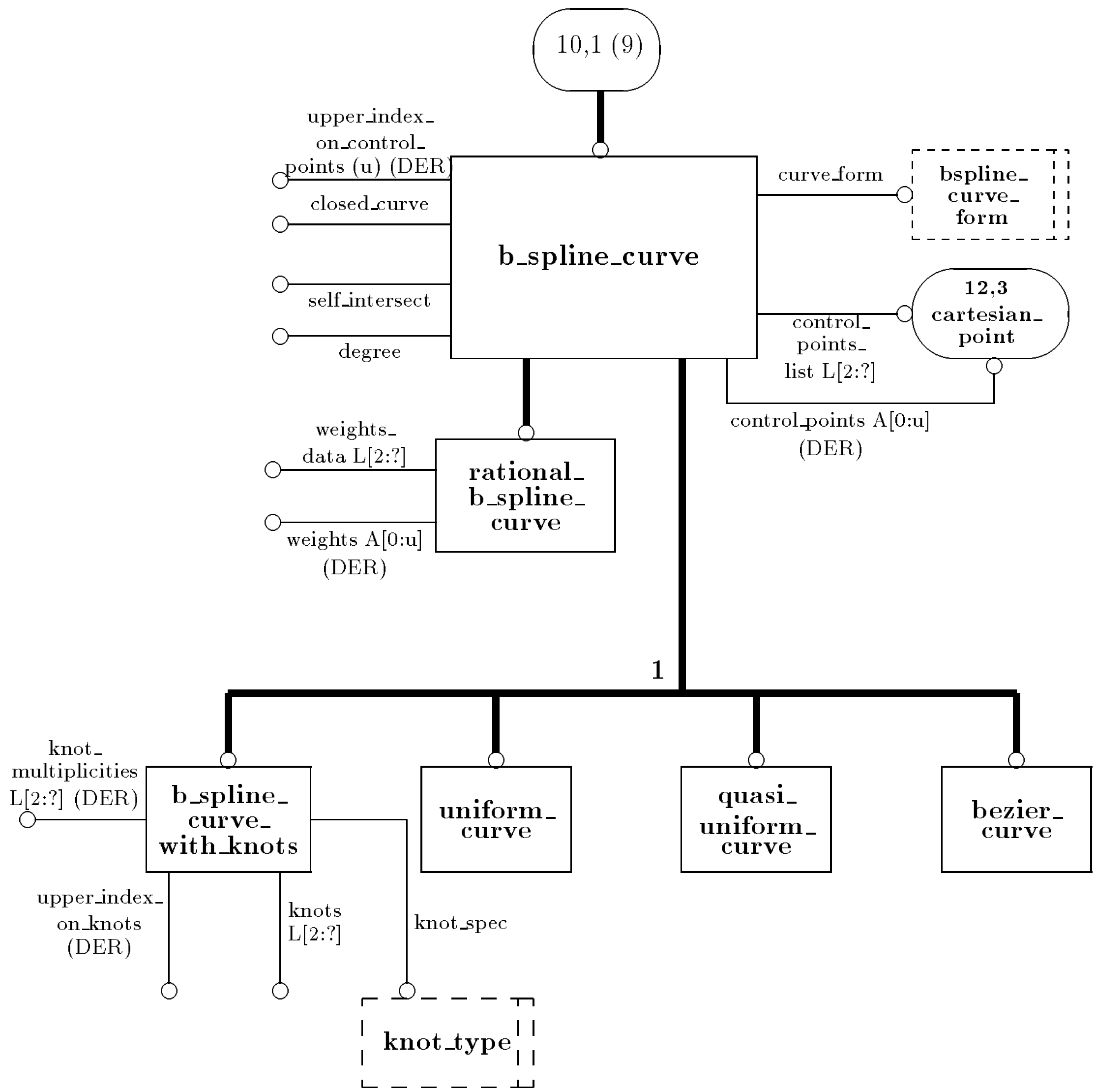

Figure G.10 – Express-G diagram, B_Spline Curve (figure 10 of 17)

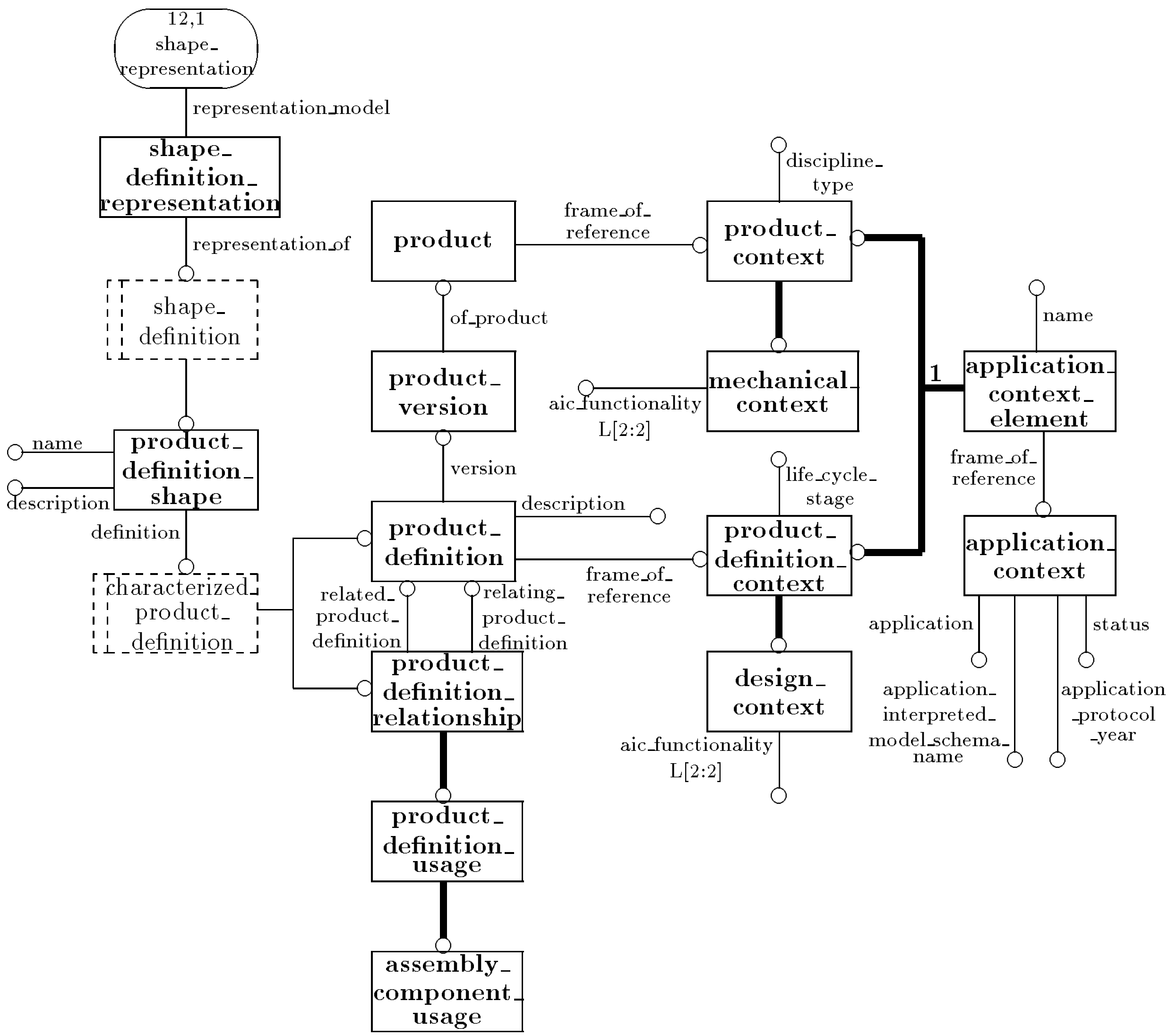

Figure G.11 – Express-G diagram, Product Structure (figure 11 of 17)

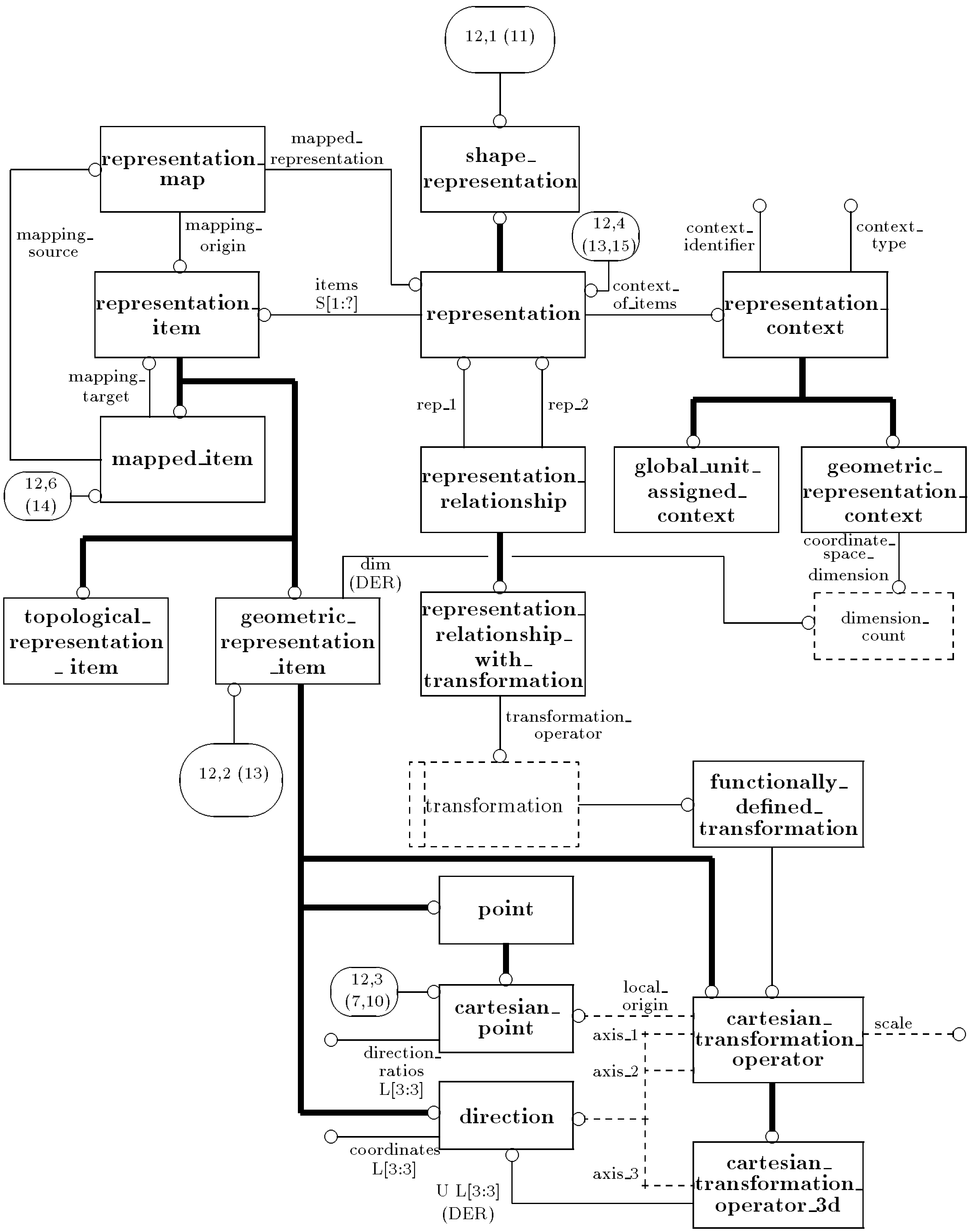

Figure G.12 – Express-G diagram, Product Structure continued (figure 12 of 17)

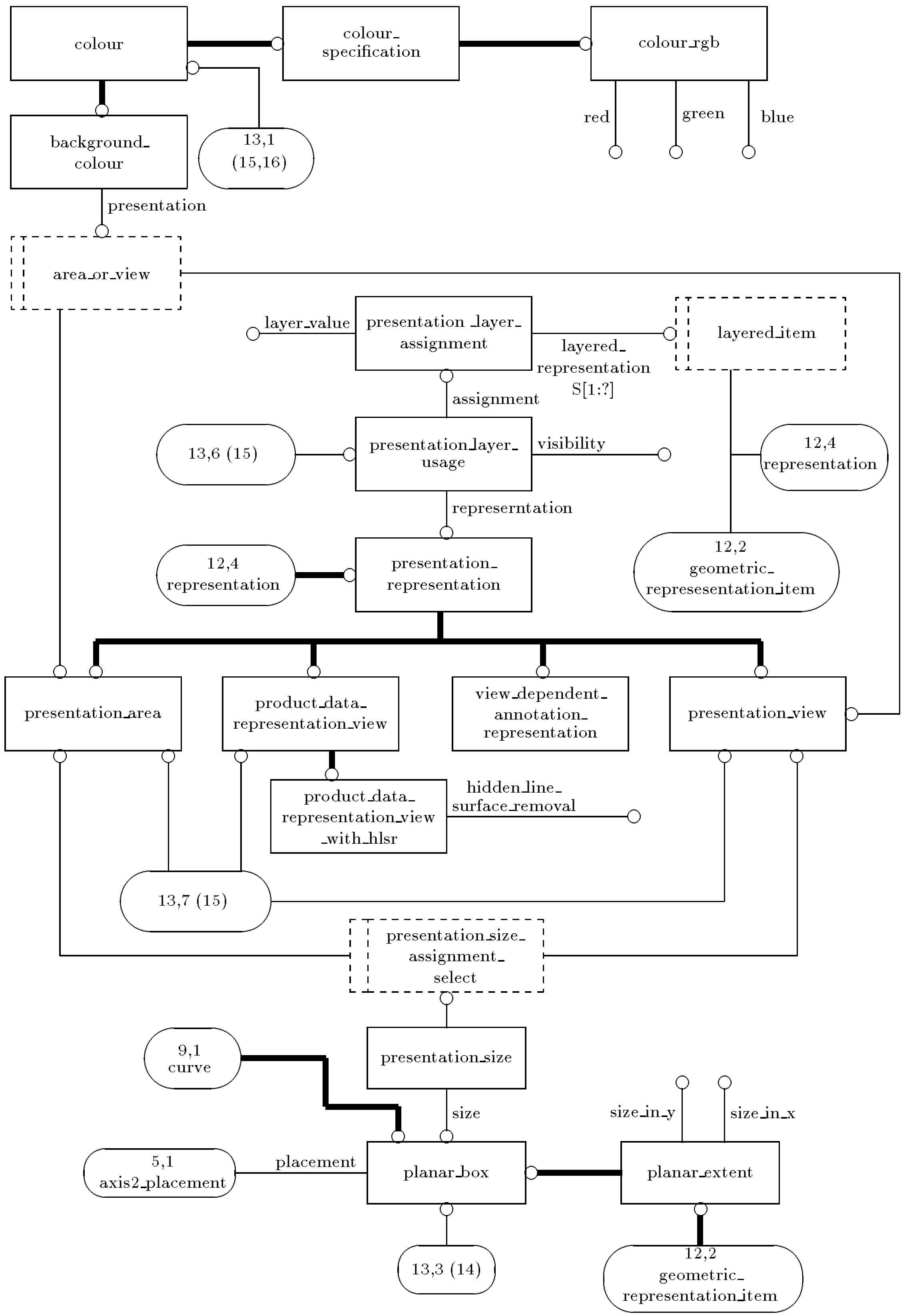

Figure G.13 – Express-G diagram, Visual Presentation (figure 13 of 17)

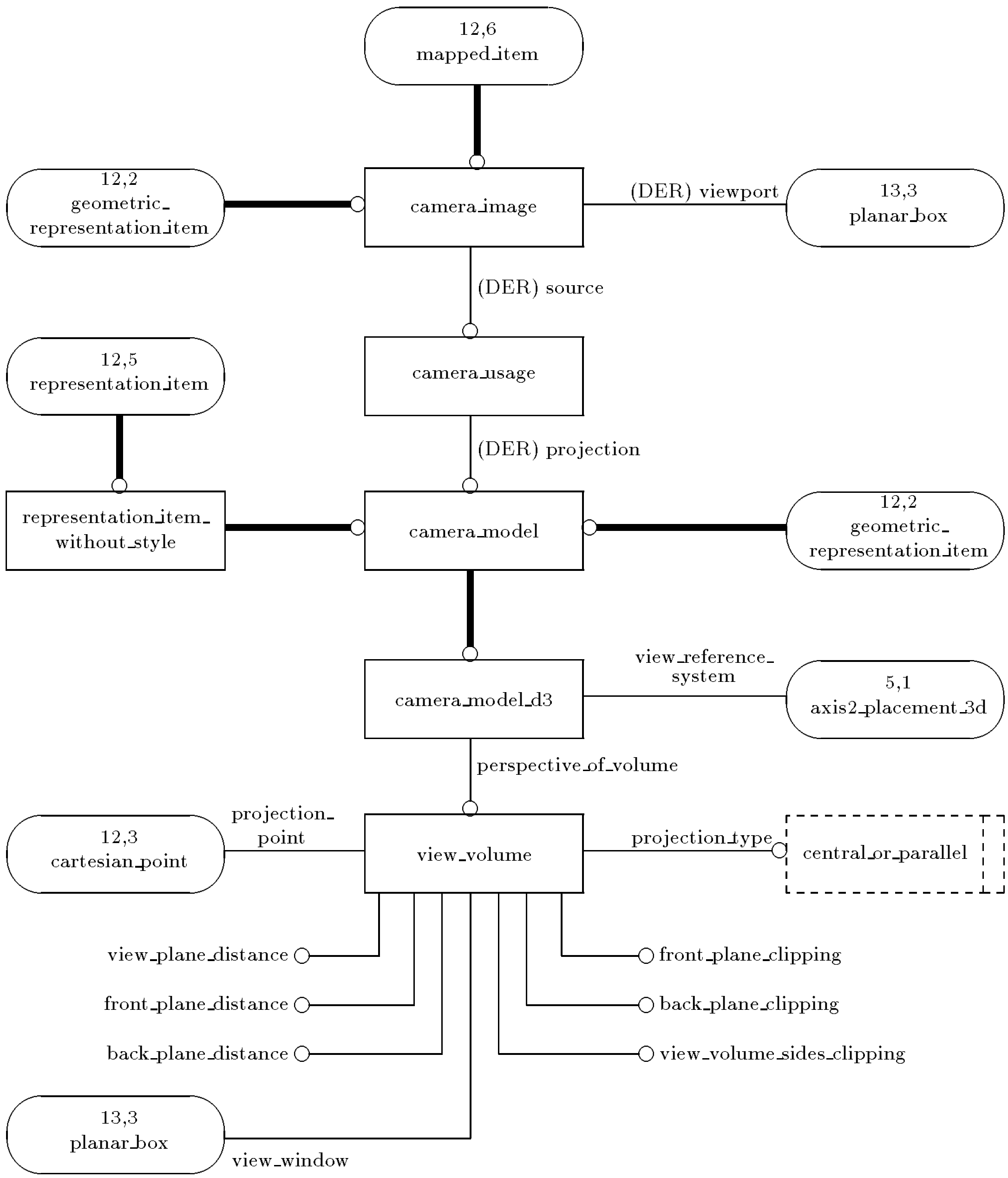

Figure G.14 – Express-G diagram, Visual Presentation (figure 14 of 17)

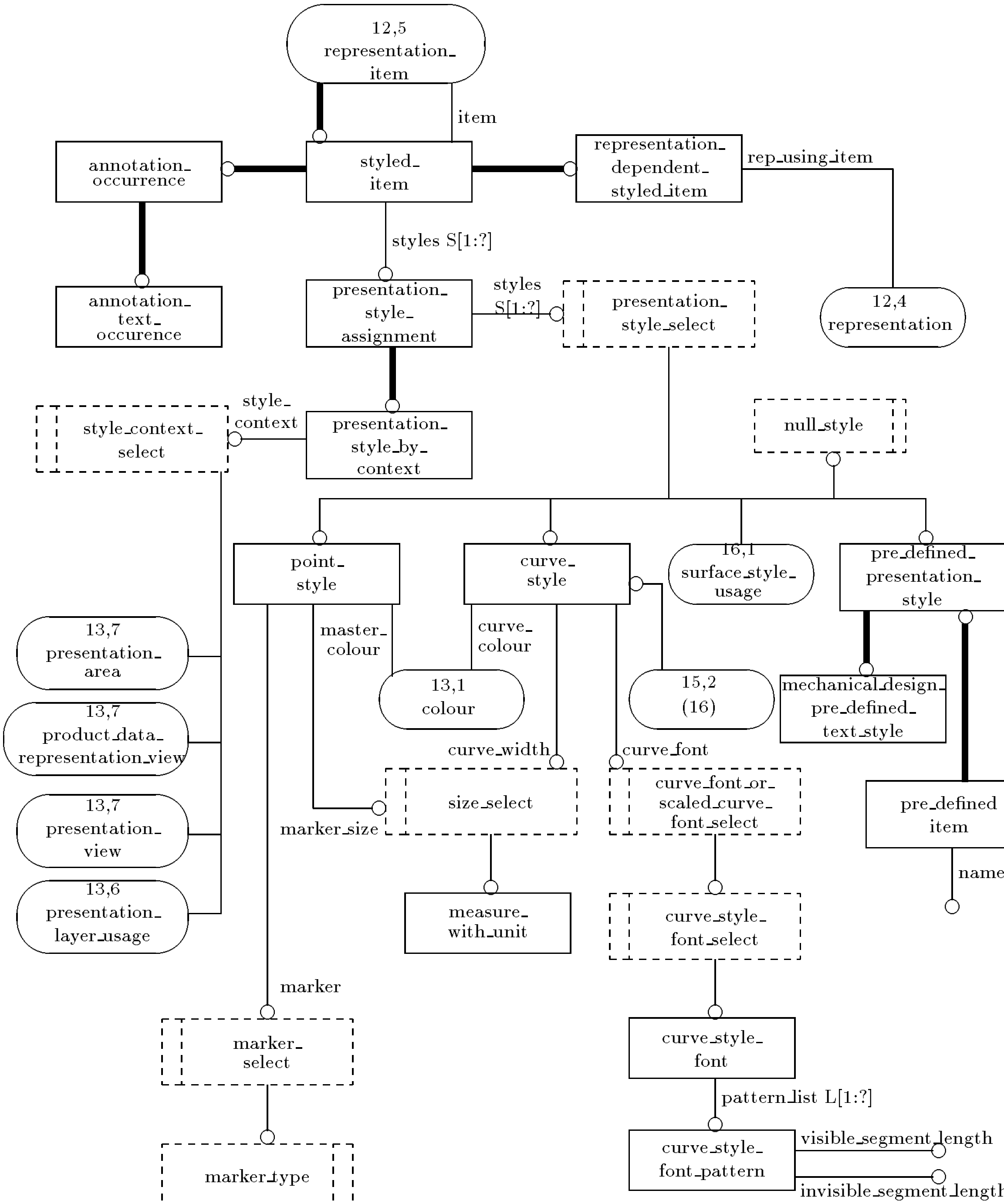

Figure G.15 – Express-G diagram, Visual Presentation (figure 15 of 17)

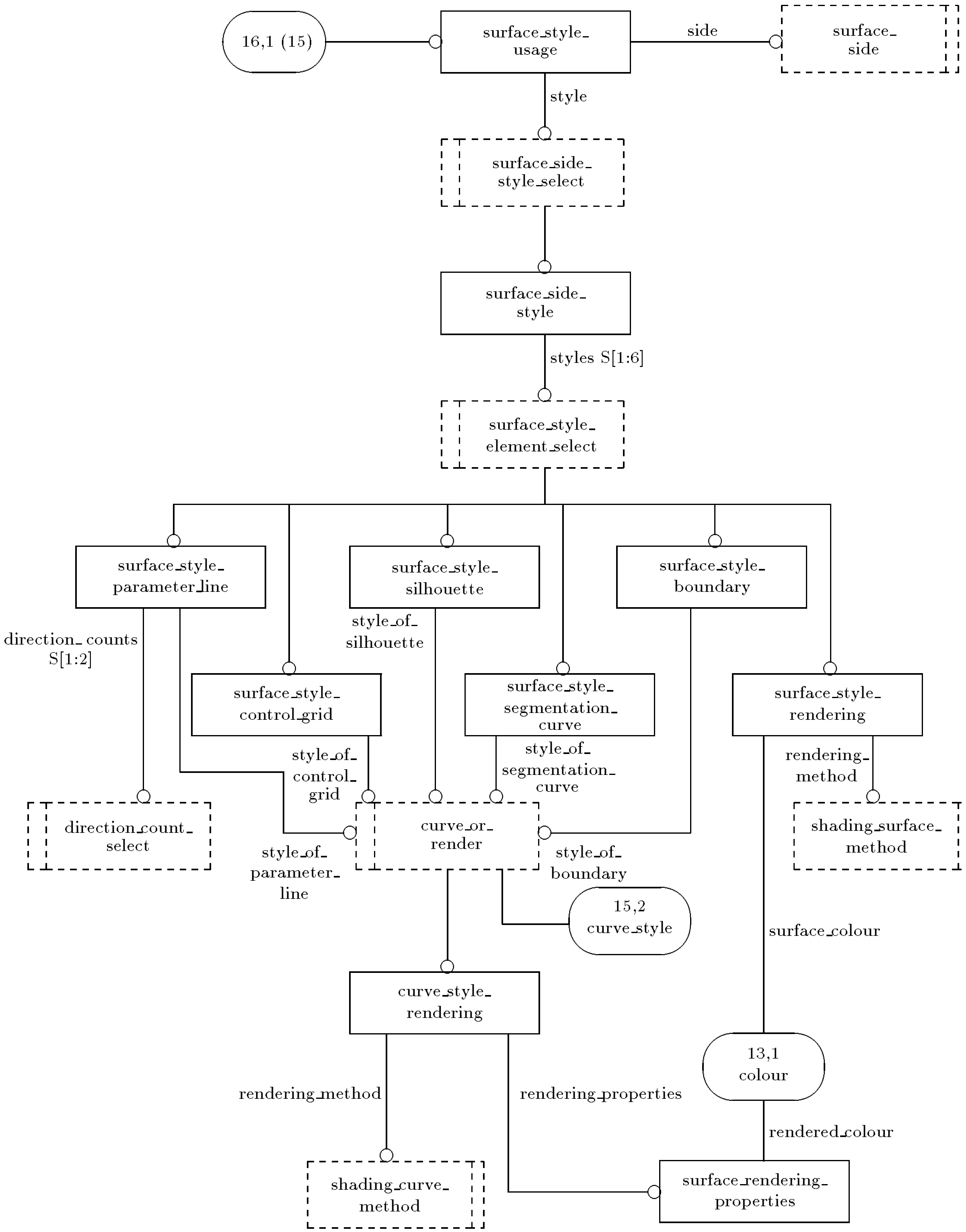

Figure G.16 – Express-G diagram, Visual Presentation (figure 16 of 17)

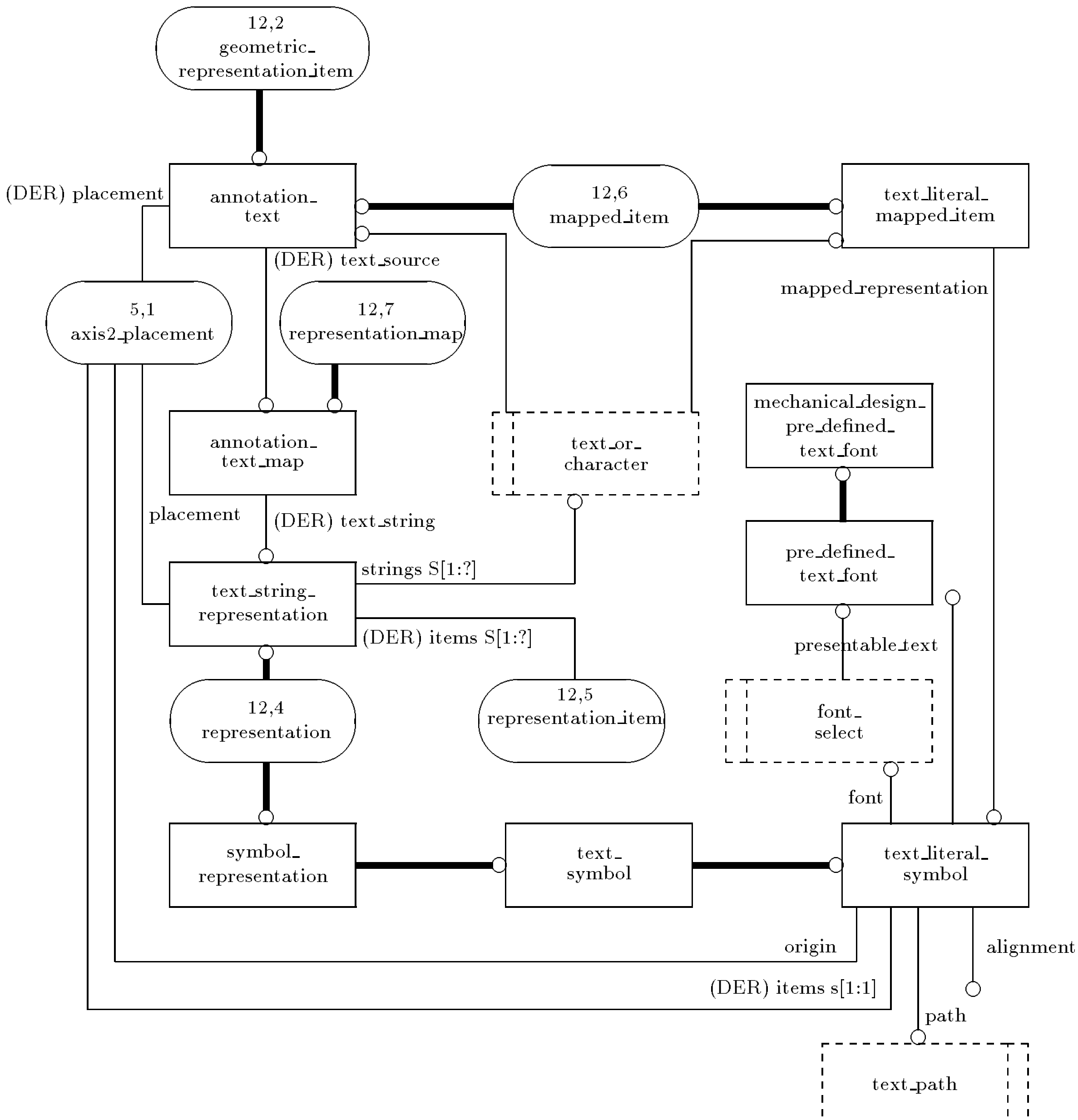

Figure G.17 – Express-G diagram, Visual Presentation (figure 17 of 17)

Annex H
(informative)

AIM EXPRESS listing.

This annex provides a listing of the table of short names and a listing of the *EXPRESS* specified in the AIM of this part of ISO 10303. No text or annotation is included. This annex is provided only in computer-interpretable form.

NOTE – The information provided on this diskette is informative; the normative text is that contained in the body of this part of ISO 10303.

Annex J
(informative)

Bibliography

[1] **G. Farin.** 'Curves and Surfaces for Computer Aided Geometric Design', Academic Press, 1988.

[2] **Bartels, Beatty and Barsky.** 'Splines in Computer Graphics and Geometric Modelling', Morgan Kaufman, 1987

[3] **Initial Graphics Exchange Specification (IGES) version 5.1** National Institute of Standards and Technology, 1991.

[4] **P. R. Wilson.** 'Euler Formulas and Geometric Modeling', IEEE Computer Graphics & Applications, Vol 5, No 8, pp. 24-36, August 1985

[5] **E.G.Schlechtendahl (Ed.),** 'Specification of a CAD*I neutral file for CAD Geometry, Version 3.3', ISBN 3-540-50392-7

[6] **W.Weick** 'Application Protocol for mechanical design using B-reps, CADEX version 4.0', Jan.92

[7] **E.G. Schlechtendahl; W.Weick.** Esprit Contributions to the Exchange of CAD models', CADCAM Yearbook 1990

Annex K
(informative)

Technical Discussions

K.1 geometric shape description alternatives

The different geometric shape descriptions in Figure E.1 distinguish themselves in model criteria like the quality of the completeness, conciseness, degrees of freedom, complexity. Dependent upon the application one or other geometric shape representation will satisfy the application requirements. In this application protocol only volume based design using B-rep models is supported.

Volume based Model Design:

Volume based design is applied to the design process for parts which have complex topology, as well as complex geometry.

Volume based design is applied in industry using two practical solid modelling techniques. These are:

A1: Boundary representation solid modelling (B-rep).This includes facetted or polyhedron B-rep models.

A2: Constructive Solid Geometry modelling (CSG).

A B-rep solid model can capture the complete topological description of different shape aspects of a part. It contains geometric surfaces and curves of its boundaries in an explicit representation. The individual elements of a B-rep model may be associated with additional model information, such as dimensional tolerance values.

The CSG solids contain boolean composition trees of basic volume primitives. The explicit topology of a part, modelled with CSG can only be obtained when the CSG tree is evaluated.

K.2 Known Issues

The following technical issues arose during the development of this application protocol.

K.2.1 Face outer bound

For some types of face surface geometry (e.g. a cylindrical surface) the face may have two or more boundaries without the possibility of identifying an 'outer boundary'. Possible resoloutions which were considered to this issue were to totally exclude all usage of the specialised **face_outer_bound** subtype, to make the usage entirely optional, or, to require this designation when appropriate.

Resolution:
In cases where the face surface is planar the outer bound can always be identified and this AP therefore requires the usage of **face_outer_bound** in the definition of all faces of a **facetted_brep**.

For other types of B-rep the conformance requirements should specify the designation of an outer bound when appropriate.

K.2.2 Complexity of geometry

Part 42 defines many types of curve and surface, not all of these are implemented in the current generation of 3D modelling systems. What criteria should be used in selecting curve and surface types for B-rep level 3 implementations?

Resolution:
In order to facilitate exchange between the largest number of modelling systems the selection was based upon the set of curve and surface types which permit the precise representation of complex shapes but which are directly defined. B-spline curves and surfaces are included but all forms of indirect definition are exccluded. In particular offset curves, offset surfaces, parameter space curves and intersection curves are excluded.

K.2.3 Representation of facetted models

A facetted B-rep is a special case of the manifold solid B-rep in which all fces are planar. The possibility exists to represent such a model with the **manifold_solid_brep** entity which would require all vertices and edges to be explicitly defined. Alternatively the **facetted_brep** subtype could be used for this purpose, in which case all vertex and edge data is implicit.

Resolution:
Conformance class 1 (B-rep level 1) is defined using the **facetted_brep** subtype. As defined in the resource part the definition of the plane defining the geometry of each face is optional since this information can, in theory, be computed from the bounding **poly_loop**s. In practice numerical precision problems make the identification of the plane of the face difficult and the **facetted_brep_shape_representation** entity defined in this Part of ISO 10303 a plane to be defined for each face.

K.3 Physical file example

A simple example of the physical file implementation of a B-rep model is included below as an informative illustration.

```
ISO-1030-21;
HEADER;
FILE_DESCRIPTION(('BREP-file', 'compatible with DIS P41,P42,P43'),
                 'BREP VERSION Feb. 1993', 'updated 15-March-1993'),
                  'BREP_AP conformance class 2, rectangular box 100/50/25');

FILE_NAME('box100-50-25.stp', '1993-03-03T18:11:01',
         ('Hermann Ruess, Peter Schild'),
         ('HEWLETT-PACKARD GmbH, MDD R&D',
          'Herrenberger Str. 130, 7030 Boeblingen, Germany'),
          'STEP 1.x (AP 204)', 'Preprocessor Version: handcrafted ',
          'Precision Engineering ', 'Hermann J. Ruess');
FILE_SCHEMA(('part_204_brep_product_schema'));
ENDSEC;

DATA;

#1 = APPLICATION_CONTEXT('status-label', 'AIM-schemaname-label',
                          AP-yearnumber, 'appl-text');
#3 = PRODUCT_CONTEXT('name-label', #1, 'disciplinetype-label');
#4 = PRODUCT(id-identifier, 'name-label', 'description-text', #3);
#5 = PRODUCT_VERSION(id-identifier, 'descript-text', #4);
#6 = PRODUCT_DEFINITION_CONTEXT('name-label' ,#1, 'life_cycle_stage-label');
#7 = PRODUCT_DEFINITION('descript-text', #5, #6);
#8 = PRODUCT_DEFINITION_SHAPE(#7);

#17 = CARTESIAN_POINT((100.00000000000, 0.00000000000, 0.00000000000));
#18 = VERTEX_POINT(#17);
#19 = CARTESIAN_POINT((100.00000000000, 50.00000000000, 0.00000000000));
#20 = VERTEX_POINT(#19);
#14 = CARTESIAN_POINT((100.00000000000, 0.00000000000, 0.00000000000));
#15 = DIRECTION((0.00000000000, 1.00000000000, 0.00000000000));
#16 = LINE(#14, #15);
#21 = EDGE_CURVE(#18, #20, #16, .T.);
#97 = ORIENTED_EDGE(*,*,#21, .T.);
#69 = CARTESIAN_POINT((100.00000000000, 50.00000000000, 25.00000000000));
#70 = VERTEX_POINT(#69);
#92 = CARTESIAN_POINT((100.00000000000, 50.00000000000, 0.00000000000));
#93 = DIRECTION((0.00000000000, 0.00000000000, 1.00000000000));
#94 = LINE(#92, #93);
#95 = EDGE_CURVE(#20, #70, #94, .T.);
#96 = ORIENTED_EDGE(*,*,#95, .T.);
#55 = CARTESIAN_POINT((100.00000000000, 0.00000000000, 25.00000000000));
#56 = VERTEX_POINT(#55);
#73 = CARTESIAN_POINT((100.00000000000, 0.00000000000, 25.00000000000));
#74 = DIRECTION((0.00000000000, 1.00000000000, 0.00000000000));
#75 = LINE(#73, #74);
```

```
#76 = EDGE_CURVE(#56, #70, #75, .T.);
#91 = ORIENTED_EDGE(*,*,#76, .F.);
#86 = CARTESIAN_POINT((100.00000000000, 0.00000000000, 0.00000000000));
#87 = DIRECTION((0.00000000000, 0.00000000000, 1.00000000000));
#88 = LINE(#86, #87);
#89 = EDGE_CURVE(#18, #56, #88, .T.);
#90 = ORIENTED_EDGE(*,*,#89, .F.);
#98 = EDGE_LOOP((#97, #96, #91, #90));
#81 = CARTESIAN_POINT((100.00000000000, 50.00000000000, 0.00000000000));
#82 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#83 = DIRECTION((0.00000000000, 1.00000000000, 0.00000000000));
#84 = AXIS2_PLACEMENT_3D(#81, #82, #83);
#85 = PLANE(#84);
#99= FACE_BOUND(#98 ,.T.);
#100 = FACE_SURFACE((#99), #85,.T.);
#26 = CARTESIAN_POINT((0.00000000000, 50.00000000000, 0.00000000000));
#27 = VERTEX_POINT(#26);
#23 = CARTESIAN_POINT((100.00000000000, 50.00000000000, 0.00000000000));
#24 = DIRECTION((-1.00000000000, 0.00000000000, 0.00000000000));
#25 = LINE(#23, #24);
#28 = EDGE_CURVE(#20, #27, #25, .T.);
#113 = ORIENTED_EDGE(*,*,#28, .T.);
#62 = CARTESIAN_POINT((0.00000000000, 50.00000000000, 25.00000000000));
#63 = VERTEX_POINT(#62);
#108 = CARTESIAN_POINT((0.00000000000, 50.00000000000, 0.00000000000));
#109 = DIRECTION((0.00000000000, 0.00000000000, 1.00000000000));
#110 = LINE(#108, #109);
#111 = EDGE_CURVE(#27, #63, #110, .T.);
#112 = ORIENTED_EDGE(*,*,#111, .T.);
#66 = CARTESIAN_POINT((100.00000000000, 50.00000000000, 25.00000000000));
#67 = DIRECTION((-1.00000000000, 0.00000000000, 0.00000000000));
#68 = LINE(#66, #67);
#71 = EDGE_CURVE(#70, #63, #68, .T.);
#107 = ORIENTED_EDGE(*,*,#71, .F.);
#106 = ORIENTED_EDGE(*,*,#95, .F.);
#114 = EDGE_LOOP((#113, #112, #107, #106));
#101 = CARTESIAN_POINT((0.00000000000, 50.00000000000, 0.00000000000));
#102 = DIRECTION((0.00000000000, 1.00000000000, -0.00000000000));
#103 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#104 = AXIS2_PLACEMENT_3D(#101, #102, #103);
#105 = PLANE(#104);
#115 = FACE_BOUND(#114, .T.);
#116 = FACE_SURFACE((#115), #105,.T.);
#33 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 0.00000000000));
#34 = VERTEX_POINT(#33);
#30 = CARTESIAN_POINT((0.00000000000, 50.00000000000, 0.00000000000));
```

```
#31 = DIRECTION((0.00000000000, -1.00000000000, 0.00000000000));
#32 = LINE(#30, #31);
#35 = EDGE_CURVE(#27, #34, #32, .T.);
#129 = ORIENTED_EDGE(*,*,#35, .T.);

#53 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 25.00000000000));
#54 = VERTEX_POINT(#53);
#124 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 0.00000000000));
#125 = DIRECTION((0.00000000000, 0.00000000000, 1.00000000000));
#126 = LINE(#124, #125);
#127 = EDGE_CURVE(#34, #54, #126, .T.);
#128 = ORIENTED_EDGE(*,*,#127, .T.);
#59 = CARTESIAN_POINT((0.00000000000, 50.00000000000, 25.00000000000));
#60 = DIRECTION((0.00000000000, -1.00000000000, 0.00000000000));
#61 = LINE(#59, #60);
#64 = EDGE_CURVE(#63, #54, #61, .T.);
#123 = ORIENTED_EDGE(*,*,#64, .F.);
#122 = ORIENTED_EDGE(*,*,#111, .F.);
#130 = EDGE_LOOP((#129, #128, #123, #122));
#117 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 0.00000000000));
#118 = DIRECTION((-1.00000000000, 0.00000000000, 0.00000000000));
#119 = DIRECTION((0.00000000000, 1.00000000000, 0.00000000000));
#120 = AXIS2_PLACEMENT_3D(#117, #118, #119);
#121 = PLANE(#120);
#131 = FACE_BOUND(#130,.T.);
#132 = FACE_SURFACE((#131), #121,.T.);
#37 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 0.00000000000));
#38 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#39 = LINE(#37, #38);
#40 = EDGE_CURVE(#34, #18, #39, .T.);
#141 = ORIENTED_EDGE(*,*,#40, .T.);
#140 = ORIENTED_EDGE(*,*,#89, .T.);
#50 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 25.00000000000));
#51 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#52 = LINE(#50, #51);
#57 = EDGE_CURVE(#54, #56, #52, .T.);
#139 = ORIENTED_EDGE(*,*,#57, .F.);
#138 = ORIENTED_EDGE(*,*,#127, .F.);
#142 = EDGE_LOOP((#141, #140, #139, #138));
#133 = CARTESIAN_POINT((100.00000000000, 0.00000000000, 0.00000000000));
#134 = DIRECTION((0.00000000000, -1.00000000000, 0.00000000000));
#135 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#136 = AXIS2_PLACEMENT_3D(#133, #134, #135);
#137 = PLANE(#136);
#143 = FACE_BOUND(#142,.T.);
#144 = FACE_SURFACE((#143),#137,.T.)
#77 = ORIENTED_EDGE(*,*,#76, .T.);
```

```
#72 = ORIENTED_EDGE(*,*,#71, .T.);
#65 = ORIENTED_EDGE(*,*,#64, .T.);
#58 = ORIENTED_EDGE(*,*,#57, .T.);
#78 = EDGE_LOOP((#77, #72, #65, #58));
#45 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 25.00000000000));
#46 = DIRECTION((0.00000000000, 0.00000000000, 1.00000000000));
#47 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#48 = AXIS2_PLACEMENT_3D(#45, #46, #47);
#49 = PLANE(#48);
#79 = FACE_BOUND(#78, .T.);
#80 = FACE_SURFACE((#79),#49,.T.);
#41 = ORIENTED_EDGE(*,*,#40, .F.);
#36 = ORIENTED_EDGE(*,*,#35, .F.);
#29 = ORIENTED_EDGE(*,*,#28, .F.);
#22 = ORIENTED_EDGE(*,*,#21, .F.);
#42 = EDGE_LOOP((#41, #36, #29, #22));
#9 = CARTESIAN_POINT((0.00000000000, 0.00000000000, 0.00000000000));
#10 = DIRECTION((0.00000000000, 0.00000000000, 1.00000000000));
#11 = DIRECTION((1.00000000000, 0.00000000000, 0.00000000000));
#12 = AXIS2_PLACEMENT_3D(#9, #10, #11);
#13 = PLANE(#12);
#43 = FACE_BOUND(#42, .F.);
#44 = FACE_SURFACE((#43), #13,.F.);
#145 = CLOSED_SHELL((#100, #116, #132, #144, #80, #44));
#146 = MANIFOLD_SOLID_BREP(#145);

#2000= (LENGTH_UNIT() NAMED_UNIT(*) SI_UNIT(.MILLI., .METRE.));

#147 = GEOMETRIC_REPRESENTATION_CONTEXT(3)
       GLOBAL_UNIT_ASSIGNED_CONTEXT(#2000))
       REPRESENTATION_CONTEXT('3D-mm','3D Cartesian, lengths in mm'));

#148 = SHAPE_REPRESENTATION((#146), #147);
#149 = SHAPE_DEFINITION_REPRESENTATION(#148, #8);
ENDSEC;
END-ISO-1030-21;
--------------------------------------------------------------------------
```

Index